



# Databand

Streamline your DataOps



# About Us

## Company

Founded 2018, 20 Engineers

## Team

### Data and ML Enthusiasts

Data Engineering, ML Engineering, Data Science,  
Data Product

ORACLE



SAP



facebook



## Funding

Well Funded, Series A Stage, Leading VCs



## Stage

Daily production use in some of the world's most exciting  
data engineering teams

Fortune50 enterprises to 50 person startups





# Contributions

- Functional DAGs (AIP31)
- Scheduler Optimizations
- In-Process Executor

## Streamline your Pipeline Code with Functional DAGs in Airflow 2.0

 Jonathan Shir [Follow](#)  
Jul 13 · 5 min read



### Intro — AIP-31

[AIP — Airflow Improvement Proposal](#)

AIP-31 was developed collaboratively across Twitter ([Gerard Casas Saez](#)), Polidea ([Tomasz Urbaszek](#)), and Databand.ai ([Jonathan Shir](#), [Evgeny Shulman](#))

When we test `DagFileProcessor.process_file` method, we obtain the following results:

Before (**commit**):

- Count queries: 1801
- DAG processing time: 8 275 ms

After (**commit**):

- Count queries: 5
- DAG processing time: 814 ms

Difference:

- Count queries: -1 796 (-99.7%)
- Processing time: -7 461 ms (-90%)

## [AIRFLOW-6181] Add InProcessExecutor #6740

**Merged** mik-laj merged 7 commits into apache:master from databand-ai:feature/python\_executor on Dec 12, 2019

Conversation 53 · Commits 7 · Checks 0 · Files changed 9 +380 -20



nuclearpenguin commented on Dec 5, 2019 · edited by dlimberman

Contributor + @ · · ·

Make sure you have checked all steps below.

### Jira

- ✓ My PR addresses the following [Airflow Jira](#) issues and references them in the PR title. For example, "[AIRFLOW-XXX] My Airflow PR"
- <https://issues.apache.org/jira/browse/AIRFLOW-6181>

### Description

- ✓ Here are some details about my PR, including screenshots of any UI changes:

Together with guys from [Databand](#) we created a new executor that is meant to be used mainly for debugging and DAG development purposes. This executor executes single task instance at time and is able to work with SQLite and sensors.

Using this executor you can debug your DAGs from IDE

### Reviewers

	ashb	
	kaxil	
	dlimberman	
	mik-laj	
	potiuk	
	reuelie	

### Assignees

No one assigned

### Labels

[area:Scheduler/Executor](#)

### Projects

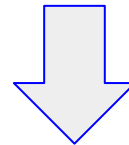
## **Problem Statement:**

In data engineering teams today, it's hard to know that pipelines are reliable.



# Data Engineering Stages

*Observability*



Identify Need

Deciding that data is worth it, realizing that analysts and scientists need help, assigning the appropriate resources.

Build

Investing in tools that get data from point A to point B, in the structure that's needed for analysis or ML.

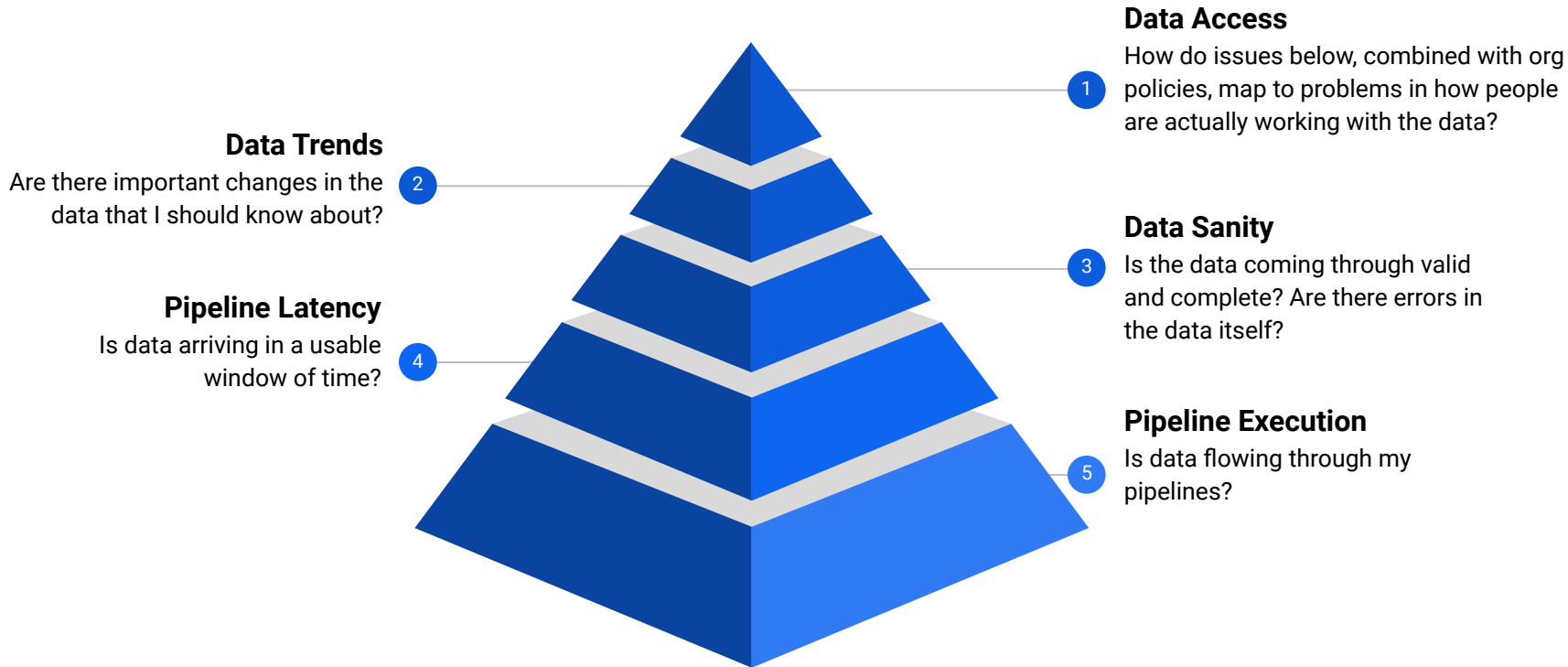
Make Sure it Works

Investing in the Ops and CI/CD practices required to ensure reliability and safe iteration.

*(Most Airflow companies here, part of why Airflow was selected)*



# DataOps Observability Pyramid of Needs



**What makes it tough?**



# Pipelines Are More Complex

Before



Informatica



Today



Google  
Cloud Composer



kubernetes



mlflow



Cloud DataProc



databricks



amazon  
EMR



python



Java



Scala



SQL



snowflake



Google  
BigQuery



amazon  
REDSHIFT



MySQL



PostgreSQL



Parquet



TensorFlow

Pandas

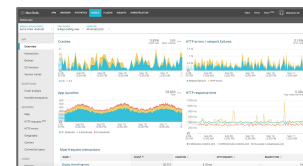
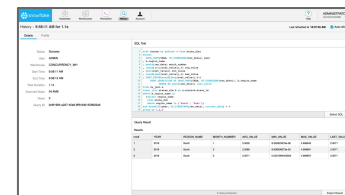
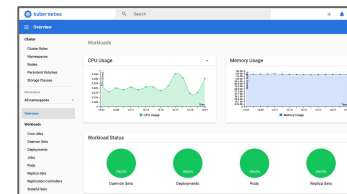
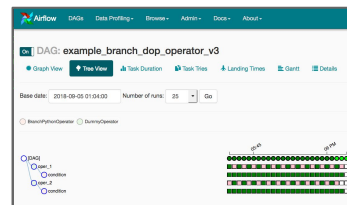
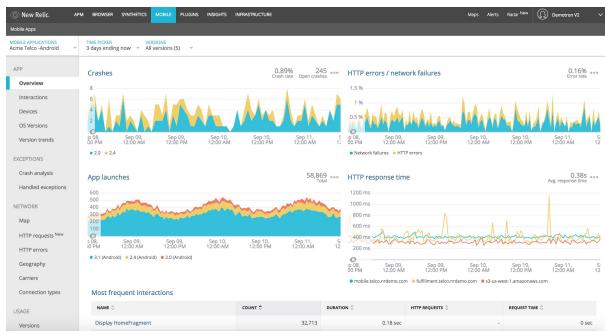


presto





# Standards Are Not Yet Defined



DevOps

DataOps



# Data Engineering Time is Precious



**2x** more job openings for data engineers than data scientists

**50%** of companies don't have enough backend data engineers

**180** hours a week spent on pipeline troubleshooting at avg company

*Stats from PulseQ&A research firm, 2018 dataops survey  
Salary figures from Indeed queries*



Databand helps data engineering teams ensure **reliable delivery** of **quality data**.

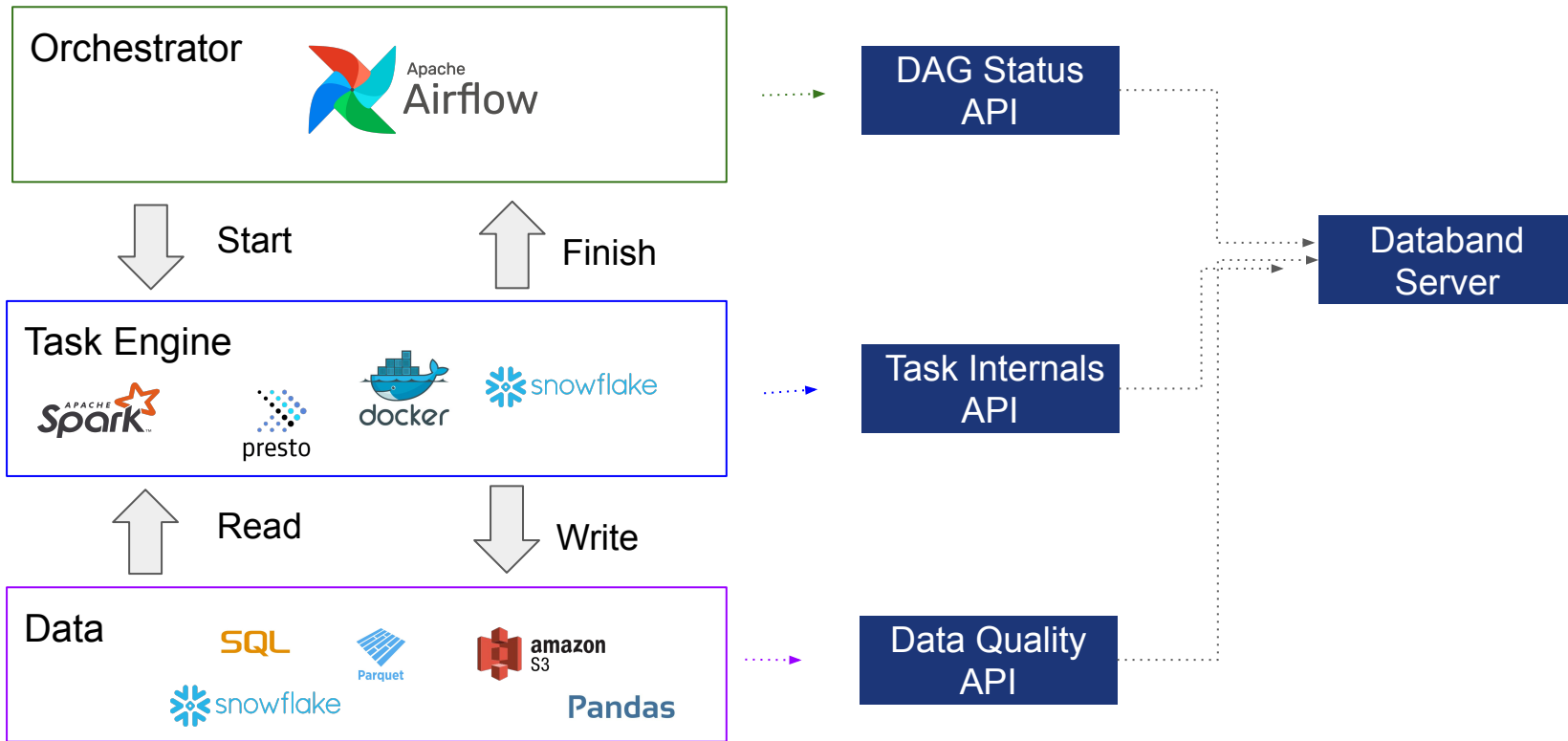
---

By providing deep monitoring on the health of data pipelines (observability).

**Collect, organize, and alert on pipeline metadata**



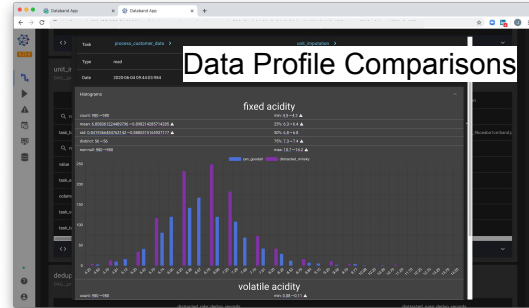
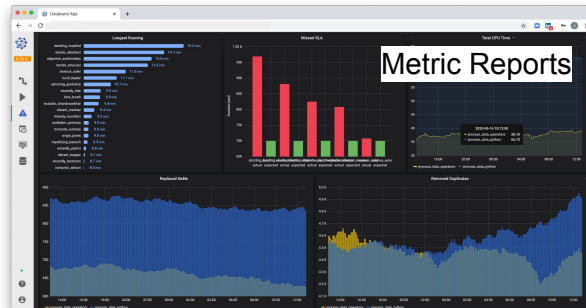
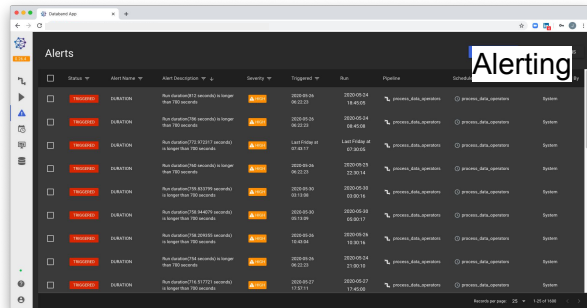
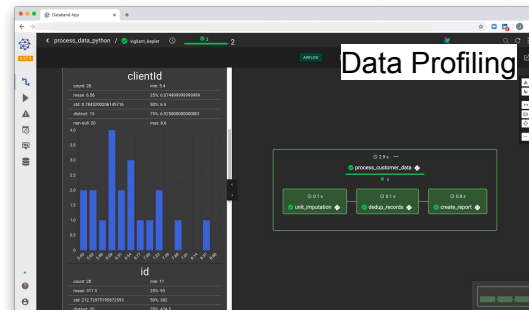
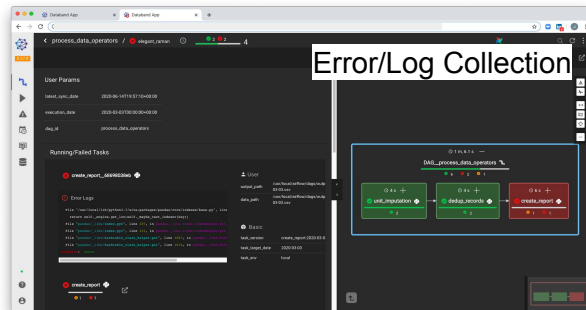
# How it Works



# Demo



The screenshot shows the Databricks workspace interface. At the top, the breadcrumb navigation indicates the path: `process_data_operators`. Below this, a toolbar contains buttons for `Run` (highlighted in blue), `Cancel`, `Download`, `Refresh`, `Help`, `More`, and `Settings`. The main content area displays a table of job runs. The table has the following columns: `Select`, `State`, `Start Time`, `Run Name`, `Progress`, `Duration`, `Scheduled Date`, `data IO mean`, `Column Count`, and `data IO std`. The table contains 10 rows of data, all with a `Completed` state. Each row shows a green progress bar at 100% completion. The `Run Name` column lists various operators such as `elapsed_time`, `printing_operator`, `elapsed_printing`, `distributed_printing`, `materialize_printing`, `reading_printing`, `writing_printing`, `insert_into_printing`, `insert_overwrite_printing`, and `projected_join`. The `data IO mean` and `data IO std` columns show values for each run. A large, semi-transparent text overlay reading "Run Tracking" is positioned in the upper right quadrant of the image.



**Contact us to learn more!**  
[contact@databand.ai](mailto:contact@databand.ai)

Always open to discussion, ideas, feedback