

A wireframe globe is centered in the background, showing latitude and longitude lines. It is rendered in a light gray color against the black background.

May 23–27, 2022

# AIRFLOW SUMMIT

Large, bright green abstract shapes are positioned on the left and right sides of the image. On the left, there is a thick, curved shape that looks like a stylized 'S' or a partial 'C'. On the right, there is a similar thick, curved shape, also partially visible. These shapes are solid green and have a smooth, rounded appearance.

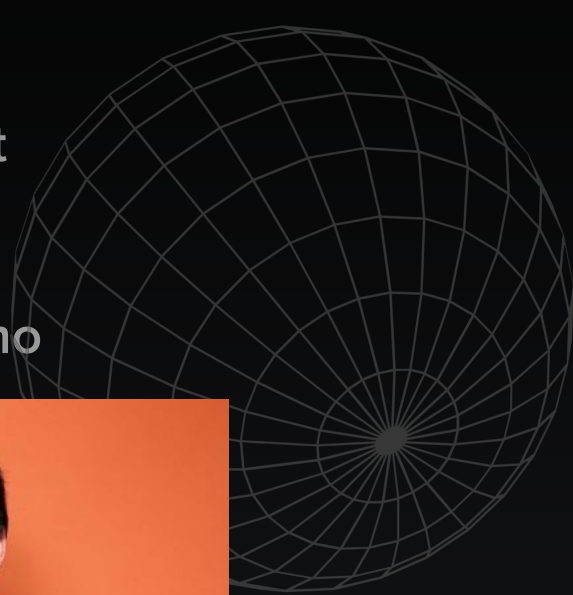
# Kyte

Scalable and Isolated Dag Development  
Experience at Lyft

Max Payton



& Paul Dittamo



# What is our scale

## Users

- Kyte - 60 WAU
- Airflow - > 350 Active users
- Over 1000 lifetime users  
(including attrition)

## DAGs

- 4000 DAGs on airflow
- 2500 Active DAGs
- Some DAGs have >1000 tasks

# Problem Statement

What Problems were we solving?

## Lots of requirements for our Airflow Instance

- Python packages
- Environment Variables
- Airflow Configurations

## Pipeline development requires production data

- Mocking data is hard
- Staging data is too random
- Statically defined data becomes out of sync with upstream

## Wide Variety of Users

- Some technical (SWE)
- Some not (Legal Compliance Teams)
- Need them all able to test DAGs

# Implementation

## What is Kyte?

### Built on existing Lyft platform

- [Lyft internal container orchestration service](#) (ML Model Training + Batch Prediction)
- Kyte container mirrors Airflow production containers

### Additional Details

- Sync DAG changes with git
- Wrapper commands to test DAGs
- No scheduler

# Key Features

## Users

1

### Isolation

Metastore Isolation  
File System Isolation  
Performance Isolation

2

### Ease of Use

Utilities Pre-installed  
Persistent State  
Jupyter Notebook Support

3

### Read access to production resources

Users write to personal schemas  
IAM roles + Envoy routing

# Key Features

## Platform/Support

4

### **Ease of Support**

Remote Development Environment =  
100% Visibility  
+  
Commoditized Instances

5

### **Auditable**

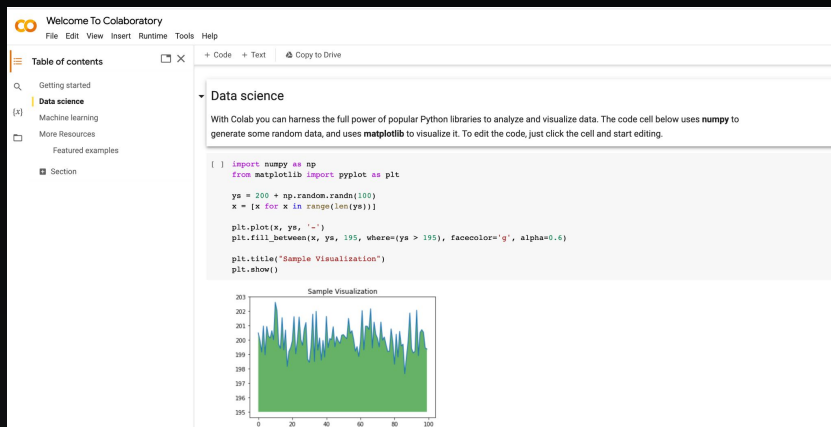
Given the access to production  
data, it's important to have a trace  
of the work

6

### **Historical use case support**

Build the new while supporting  
the old

# Inspired by Google Collab and Polynote



Google Colaboratory interface showing a code cell and its output. The code cell contains Python code for generating random data and visualizing it using matplotlib. The output is a line plot titled "Sample Visualization" showing a green area under a blue line.

```
( ) import numpy as np
from matplotlib import pyplot as plt

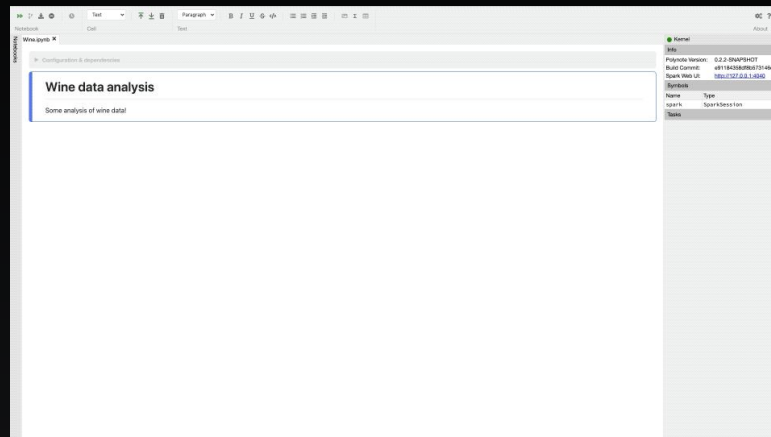
ys = 200 + np.random.randn(100)
x = [x for x in range(len(ys))]

plt.plot(x, ys, '-')
plt.fill_between(x, ys, 195, where=(ys > 195), facecolor='g', alpha=0.6)

plt.title("Sample Visualization")
plt.show()
```

Sample Visualization

x	ys
0	200.0
10	205.0
20	195.0
30	210.0
40	198.0
50	202.0
60	196.0
70	208.0
80	199.0
90	201.0
100	197.0



Jupyter Notebook interface showing a code cell and its output. The code cell contains Python code for generating random data and visualizing it using matplotlib. The output is a line plot titled "Wine data analysis" showing a green area under a blue line.

```
( ) import numpy as np
from matplotlib import pyplot as plt

ys = 200 + np.random.randn(100)
x = [x for x in range(len(ys))]

plt.plot(x, ys, '-')
plt.fill_between(x, ys, 195, where=(ys > 195), facecolor='g', alpha=0.6)

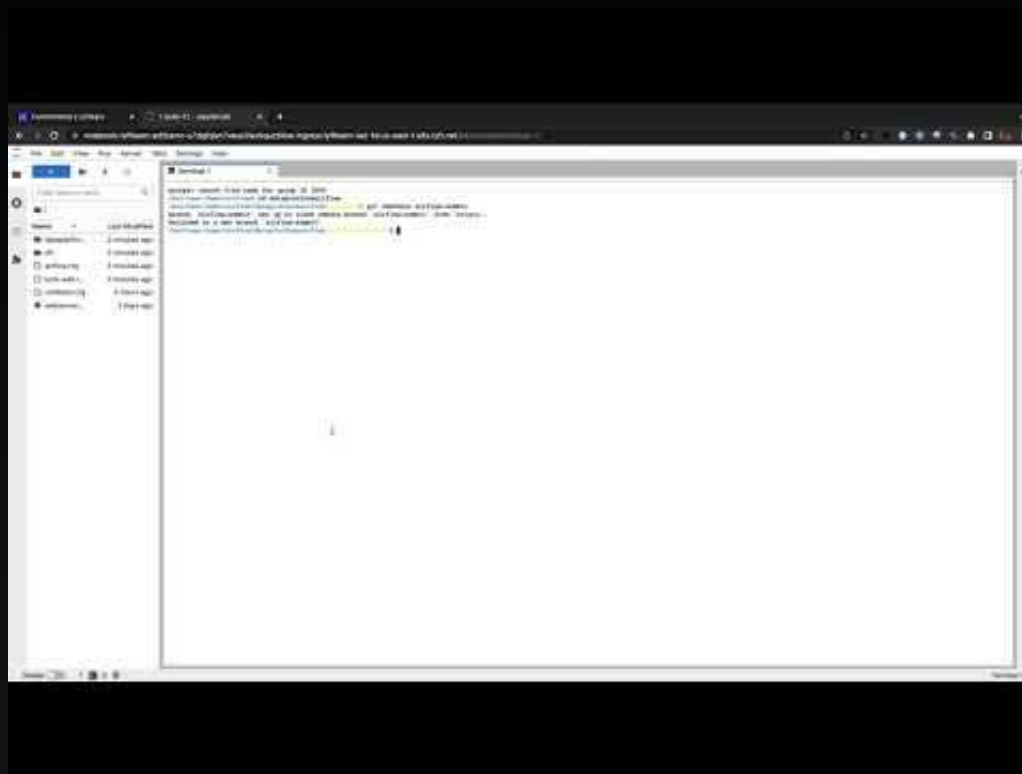
plt.title("Wine data analysis")
plt.show()
```

Wine data analysis

x	ys
0	200.0
10	205.0
20	195.0
30	210.0
40	198.0
50	202.0
60	196.0
70	208.0
80	199.0
90	201.0
100	197.0



# Demo



# Learnings

What did we learn?

## Users don't love Github as a syncing mechanism

- Users might be unfamiliar with Github syntax
- Uncomfortable committing incomplete code
- Some context switching between environments

## Varied preferences for developing locally versus remotely

- Jupyter incomplete as IDE
- Remote environment introduces lag
- Setting up IDE can be a cost

## Testing failure cases can be difficult

- SLA misses
- Upstream failures
- BranchPythonOperator cases

# Next Steps

## Future Work

### Testing Support

- Failure cases
- Integration testing of DAGs

### Connect the Notebook Kernel locally

- Improve local development

### Linting/Static Analysis

- Already some work for this open source
- Errors introduced when SQL is copied over and parameterized
- Blessed DAG structure

# The Dream

Northstar

**As Jupyter Notebooks became to  
Data Analysis, Kyte will become to  
Airflow dag development**

