Looking ahead: What comes after Airflow 2.0?

Aizhamal Nurmamat kyzy
Ash Berlin-Taylor
The Road so Far...
Airflow Timeline

- **2014**: Started at AirBnB
- **2015**: Open Sourced
- **March 2016**: Donated to Apache Software Foundation
- **December 2018**: Graduated as a top-level project
- **July 2020**: Inaugural Airflow Summit
- **December 2020**: Airflow 2.0 released
- **July 2021**: Airflow Summit V2
Airflow Today

Active
- Top 5 Apache project repositories by commits in 2020

Collaborative
- +1,500 code authors

Supportive
- Continued active releases for v1 and v2

Popular
- Hundreds of companies using Airflow in the wild
Downloads of Airflow packages per week

- v2
- v1

Source: pypistats.org
Community > Code
Airflow Summit 2020

~6000 registrations
X Sponsors
31 Committers
21 PMC members

Airflow Summit 2021

~9000 registrations
XX sponsors
44 Committers
24 PMC members
Transition
Apache Airflow 2.2 and beyond
We build our computer (systems) the way we build our cities: over time, without a plan, on top of ruins

— Ellen Ulman
Roadmap: A possible future
Making DAGs a joy to write
Airflow should be the go to orchestrator for every data workflow job
Airflow should be easier to operate confidently
Roadmap Concepts

- Making DAGs a joy to write
- Airflow should be the go-to orchestrator for every data workflow job
- Airflow should be easier to operate confidently
Airflow 2.2
AIP-39: Run DAGs on customizable schedules
Why AIP-39
AIP-40: Any operator can "defer" itself
Why AIP-40
The near future
Where's the new UI?

AIP-38

Adding missing API endpoints first

Don't want to ask people to not contribute new UI features

Nor do we want to chase a moving target
airflowctl: CLI over the API
Untrusted workers
DAG/task lifecycle hooks and easier notifications
task = MyOperator(
    task_id = "something",
    on_failure_callback=send_slack_message(
        channels=['#data-ops'],
        mentions=['@ash'],
    ),
)

@task(on_failure_callback=[send_slack_message(), send_email]
def my_task():
    ...

Dynamic DAGs
@task
def get_files_from_s3():
    ...

my_files = get_files_from_s3()
s3_delete_files = S3DeleteObjectsOperator.partial(
    aws_conn_id="my-aws-conn-id",
    bucket="my-bucket"
).map(key=my_files)
@dag
def my_dag(markets: list[str], campaigns: dict[str, list[int]]):
    @task
def ingest(market):
        ...
    @task
def calculate_roi(market, campaign):
        ...
    @task
def aggregate_rois(market, campaign_rois):
        total = 0
        n = 0
        for campaign_roi in campaign_rois:
            n += 1
            total += campaign_roi
        return campaign_roi/total

data = ingest.map(markets)
rois = calculate_roi.map(market, data)
stats = aggregate_rois(market, rois)
A better cross-DAG story
Looking further ahead
Event triggered DAGs
DAG versioning
Easier DAG deployment
Streaming
New concept: a Data object
Better support for Machine Learning
Airflow 3.0?