

From Cron to Apache Airflow

A Startup Story

Adam Boscarino, 2020-07-13

Who am I?

- Data Engineer at Devoted Health
- Previously worked at DigitalOcean, Fitbit, Carbonite
- Airflow user for ~4 years
- GitHub & Twitter: ajbosco



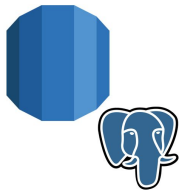
What is Devoted Health?

- A next generation Medicare Advantage health insurer in the United States
- Founded in 2018, first enrolled members in 2019
- Offers a Clinical Services solution (Devoted Medical Group)
- Built on homegrown Devoted Tech Platform

**“ TO DRAMATICALLY IMPROVE HEALTHCARE
FOR SENIORS IN THE UNITED STATES -- CARING
FOR EVERYONE LIKE THEY ARE MEMBERS OF
OUR OWN FAMILY ”**

Devoted Health Data Platform, January 2019

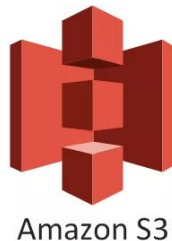
Source Data



Workflows



Storage/Data Lake



Data Warehouse



Reporting/BI



Devoted Health Data Platform - Successes

- It did its job
- Successfully launched new health plan
- Supported key business operations and workstreams
- Powered all internal reporting



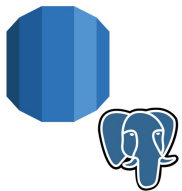
Devoted Health Data Platform - Problems

- No task dependencies
- Undetected system downtime
- Onboarding new developers
- Environment parity
- Unsure of data quality



Devoted Health Data Platform, May 2019

Source Data



Workflows



NEW!

Storage/Data Lake



Amazon S3

Data Warehouse



NEW!

Reporting/BI



Devoted Health Data Platform - Problems

- **No task dependencies**
- Undetected system downtime
- Onboarding new developers
- Environment parity
- Unsure of data quality

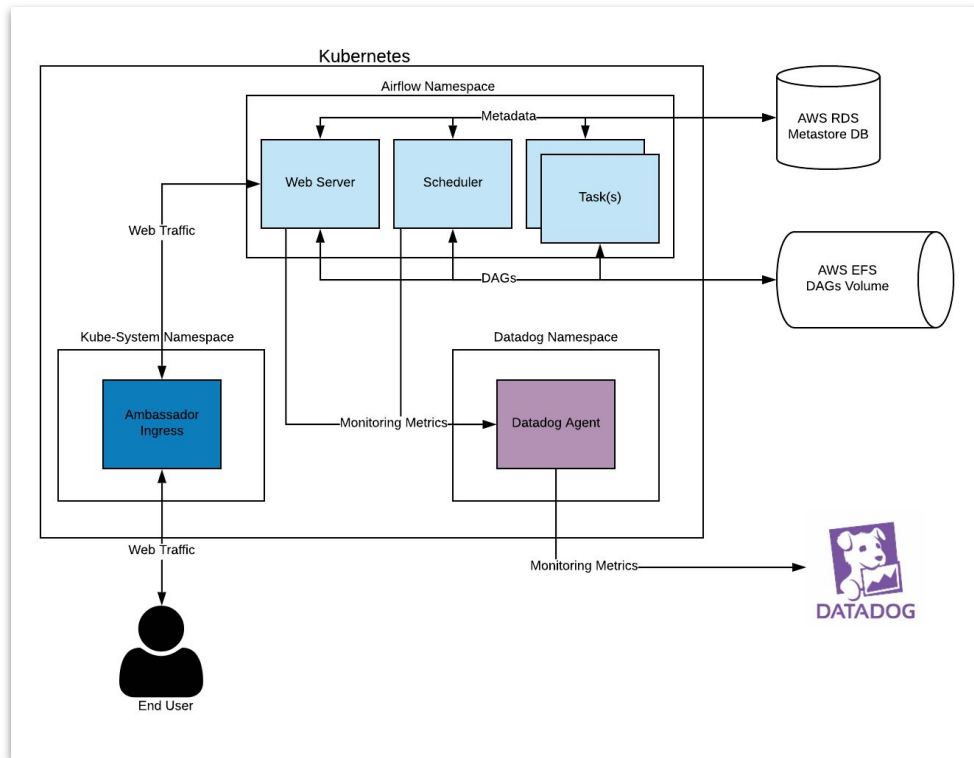


Solution: Deploying Airflow



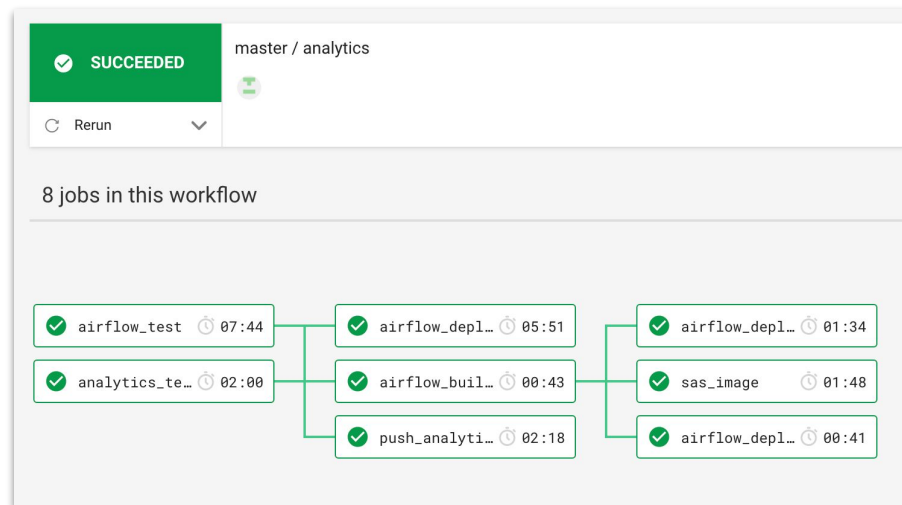
Apache Airflow Deployment

- Kubernetes
 - Orchestrates Airflow services
 - Kubernetes Executor
- Helm
 - Kubernetes Package Manager
 - Describes Kubernetes resources
 - Official Helm Chart
- Terraform
 - Infrastructure as Code
 - Used to deploy Helm chart to Kubernetes clusters



DAG Deployment

- DAGs are stored on AWS EFS
 - Mounted to each Airflow pod in Kubernetes
- DAGs are pushed from GitHub to AWS EFS via CircleCI
 - No manual intervention
 - Many deployments every single day



Devoted Health Data Platform - Problems

- No task dependencies
- **Undetected system downtime**
- Onboarding new developers
- Environment parity
- Unsure of data quality

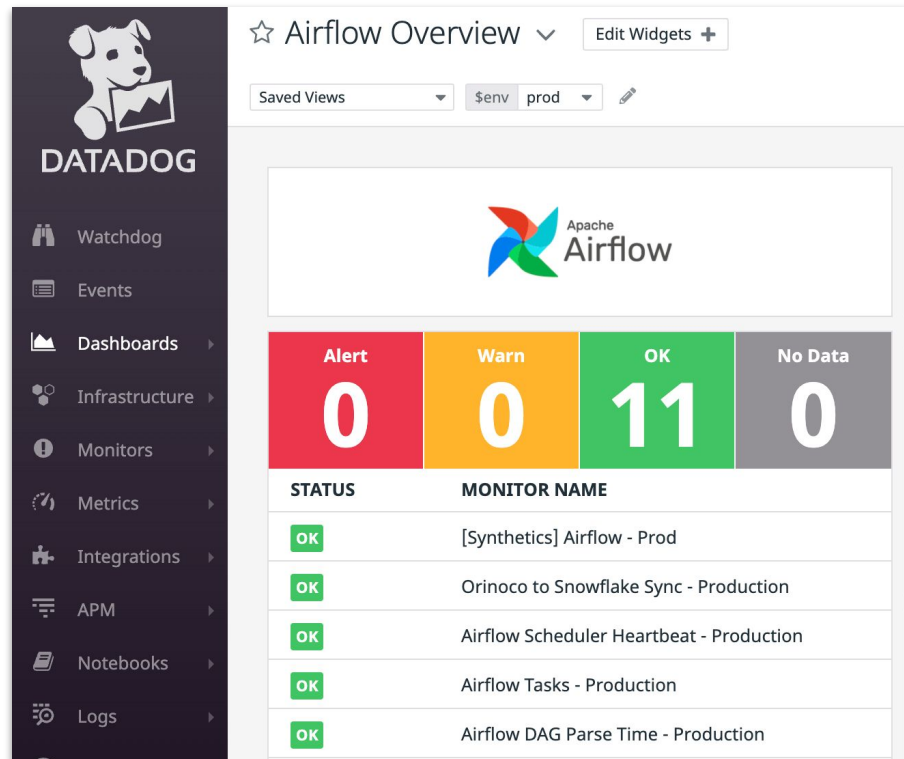


Solution: Monitoring Airflow



Monitoring Airflow

- Kubernetes Liveness & Health Checks
 - Monitor /health endpoint
 - Monitor Scheduler health
 - Restart services if in bad state
- Datadog Monitors
 - Alert on-call engineer via OpsGenie and Slack
 - Airflow is not running
 - No DAGs have completed in last 2 hours
 - CPU/Memory Usage has spiked



Monitoring DAGs

- OpsGenie alerts sent to DAG Owner (and Slack)
- DAG owners are responsible for resolving non-infrastructure failures
- Alerting is “built-in” to every DAG

Alerts

☒ See all alerts ☐ Select

Saved searches

PREDEFINED


All

☐ #4305

P3

Airflow - analytics failed in prod

x1

 Data Science

Devoted Health Data Platform - Problems

- No dependency management
- Undetected system downtime
- **Onboarding new developers**
- Environment parity
- Unsure of data quality



Solution: YAML DAG Builder



YAML DAG Builder

- Internal library to simplify and standardize DAG development.
- Abstraction on top of Airflow.
- Developers only write a DDL query and transformation in SQL or Python.
- No prior Airflow knowledge required.
- Similar structure in all DAGs makes switching teams less painful and simplifies debugging DAGs.
- Data Engineer team can bolt on additional features (alerting, monitoring, testing, etc.)

```
dag: 'example_dag'
owner: 'Data Science'
schedule: '30 */4 * * *'

prep_schema: 'staging'
final_schema: 'warehouse'
base_path: 'warehouse/example_dag/'

tasks:
  dim_table:
    config_type: 'SqlTask'
    ddl: 'ddl/dim_table.sql'
    sql: 'extractors/dim_table.sql'

  fact_table:
    config_type: 'PythonTask'
    ddl: 'ddl/fact_table.sql'
    python: 'extractors/fact_table.py'
    deps:
      - dim_table
```

Devoted Health Data Platform - Problems

- No task dependencies
- Undetected system downtime
- Onboarding new developers
- **Environment parity**
- Unsure of data quality



Solution: devflow



- Internal tool that wraps kubectl, Helm, and Terraform.
- Every developer gets their own Airflow deployment on Kubernetes.
- We develop on the same stack that we run production.
- Developers do not need to know anything about the infrastructure being used.

```
DEVFLOW

=====
Run 'source with-role analyst@dev' prior to using devflow or it will not work!!!
=====

usage:
-h|help      display usage
start        start Airflow instance on dev k8s cluster
restart      restart Airflow instance on dev k8s cluster
stop         stop Airflow instance on dev k8s cluster
stop-all    stop all Airflow instances on dev k8s cluster that are older than 24 hours;
             pass a different number of hours to limit to older releases, e.g. stop-all 2
sync         sync DAGs and Plugins directories to Airflow
status       check status of Airflow instance
logs         view logs from Airflow web server
logs-scheduler view logs from Airflow scheduler
shell        open a bash shell in the Airflow instance
clone schema/table clone tables or schemas in * from Prod to your Dev database
deploy udfs   deploy latest UDFs to your Dev database *
analytics-push builds and pushes 'data-pipeline-stg' image to ECR
push-dev-image builds and pushes 'airflow-dev' image to ECR
```

Devoted Health Data Platform - Problems

- No task dependencies
- Undetected system downtime
- Onboarding new developers
- Environment parity
- **Unsure of data quality**



Solution: Testing & Validation



Testing DAGs

- Unit Tests
 - Used on Python transformations and core library code
- Integration Tests
 - Used for SQL tasks
 - Internal framework built on pytest
 - Executed against Snowflake using a test database
 - Mock tables are created and populated

➤ **fmt**

➤ **yamllint**

➤ **lint**

➤ **unit tests**

➤ **setup snowflake connection**

➤ **dags - integration tests**

Data Validation Framework

- Data validation is executed at DAG run-time
- DAGs are stopped if validation fails to prevent reporting on bad data
- Started with Check Operator
- Added internal Operators
 - Runs multiple checks with one task
 - Save invalid records to table
 - Send check values to Datadog
- Checks range from primary key validation to custom business logic

```
table_a_pk_validation:
  config_type: ValidateTask
  validation_type: 'uniqueness'
  validation_table: table_a
  validation_columns:
    - id
  deps:
    - table_a_populated

table_a_count_validation:
  config_type: ValidateTask
  validation_type: 'custom'
  validation: 'validations/table_a_count_validation.sql'
  operation: '='
  pass_value: 0
  deps:
    - table_a_populated

table_c_quality_checks:
  config_type: QualityCheckTask
  description: 'Runs all data quality checks for table C.'
  quality_checks: 'validations/table_c_validation.py'
  deps:
    - table_c_populated
```

Mission Accomplished!



Current Issues & Future Work

- Improve SQL testing!
 - Explore tools like dbt and dataform
 - Remove need for end user to know pytest
- Improve DAG Builder
 - Make standard use cases easier
- SQL Linting/Formatting
 - Enforce best practices programmatically
- KEDA Autoscaler
 - Improve task spin-up speed



Questions?

