Apache Airflow

Data Engineering Lunch - 12/07/2020
Will Angel

Linkedin: https://www.linkedin.com/in/william-angel/
Twitter: @DataDrivenAngel

Overview:

1. Presentation:
    a. What is Airflow?
    b. How does Airflow work?
    c. Why should I use Airflow instead of Cron (or Luigi?)

2. Demo:
    a. Airflow admin interface
    b. Creating workflows
    c. Scheduling workflows
    d. Monitoring and retrying workflows

What is Apache Airflow?

Airflow is an open source workflow scheduling tool.

With Airflow you can:

- Write workflows in Python (and anything you can call from python)
- Schedule and execute workflows
- Monitor workflows and log results.

How does Airflow Work?

The Airflow application reads, schedules, and executes workflows that have been defined in python files as Directed Acyclic Graphs (DAGs).

1. The Scheduler service monitors the workflow files and schedules workflows to be executed as defined by the DAG's crontab (cron table notation)
2. The Scheduler sends the workflow to an Executor service which runs the workflow.
3. The Executor sends task status updates to the scheduler and does logging, metrics, and updates task metadata.

How does Airflow work (continued)

Airflow also has a webserver based GUI that provides a great experience if you don't want to learn the entire CLI/API.

Web app GUI features include:

- Edit/trigger workflows
- Monitor individual and aggregate workflow runs (lots of well designed charts)
- Update configurations

Why not use Cron?

- So you can sleep at night.
  - Native support for logging, alerting, and retrying scheduled workflows.
  - Scalable with Kubernetes. Cron is limited to a single computer.
  - Airflow workflows can be executed on Kubernetes, Dask, or Celery.
- Airflow can be extended and has a large ecosystem of extensions

The biggest reason: Git

DAGs are defined as python files.

All files are in a single folder

Files can be version controlled.

Why use Cron - the downsides

Airflow has downsides and pitfalls like every other tool:

- Overhead - Airflow requires a compute instance, which usually will need to be larger than a machine handling the same number of tasks in Cron.
- Maintenance - Airflow is usually durable and low maintenance, but if you write custom extensions it will be as good as your custom extensions.
- Complexity - Airflow is simple at a high level, which makes good data engineering practices easier. Implementing airflow is a good time to also add alerting, logging, monitoring, ETL version control, data lineage, etc.

Core Airflow Concepts

1. Workflows: DAGs - Directed Acyclic Graphs
2. DAG Runs
3. Tasks
4. Task Instances
5. Task Lifecycle
6. Operators

Core Questions- Task Scheduling

1. How do we define a task in a way that a computer can run it?
2. How do we know a task has run successfully?
3. How should a task execution attempt handle failure?
4. How do we keep track of changes in a recurring task?

Demo!