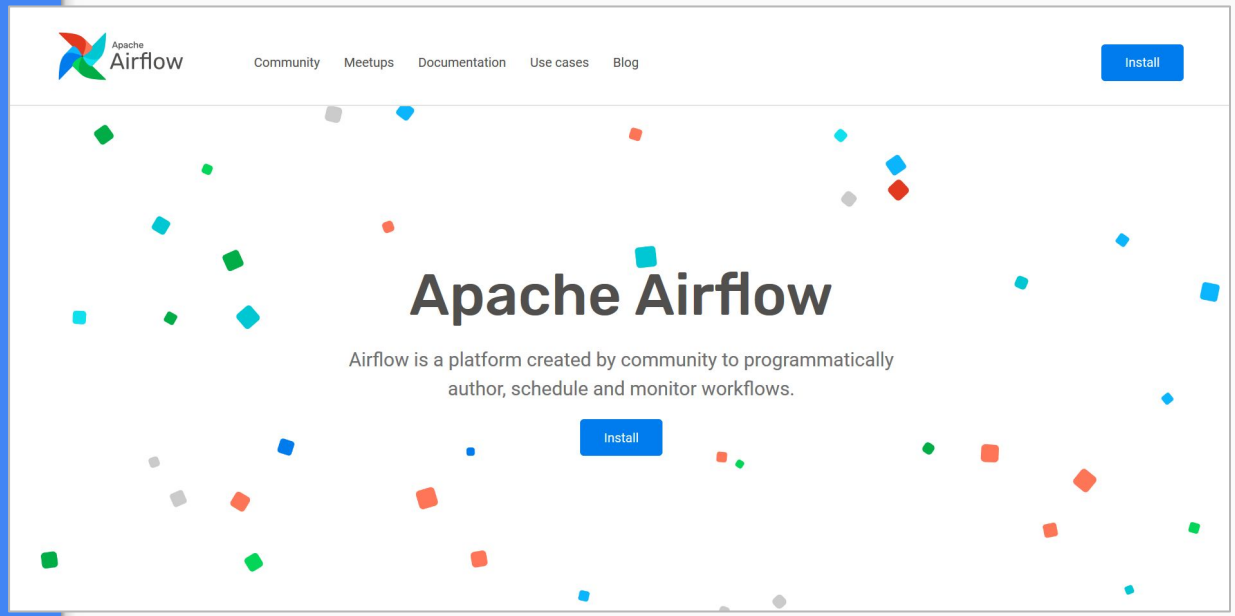
Apache
Airflow

Nosso João Carlos Martis dos jobs

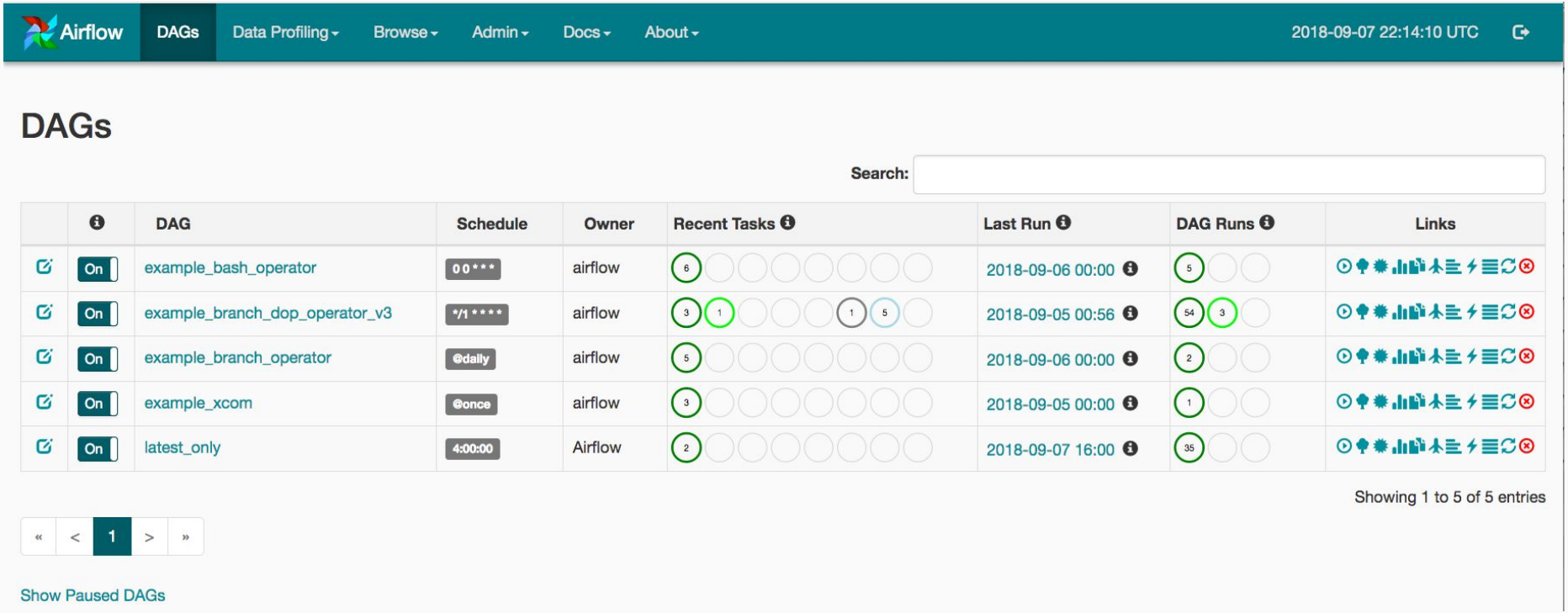
O que é?

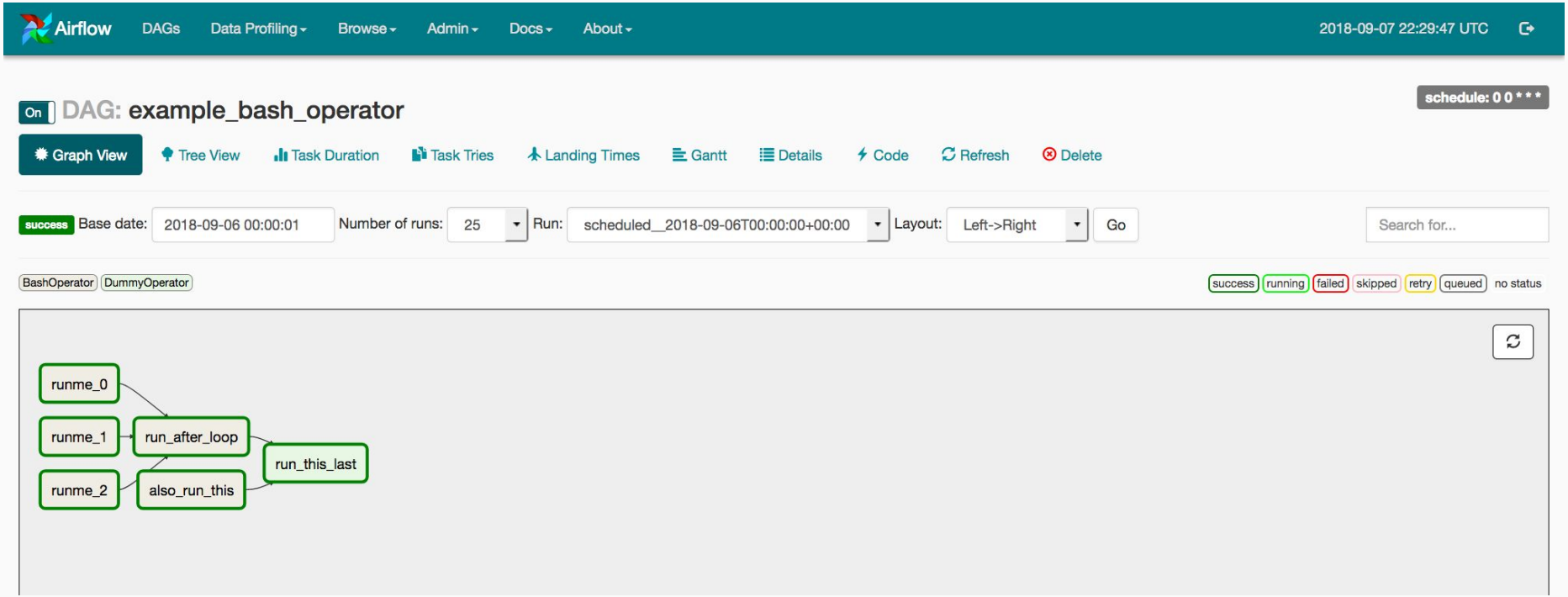
Plataforma para publicar, agendar e monitorar workflows.

- **2014:** início do projeto pelo Maxime Beauchemin at Airbnb
- **2015:** transformado em projeto de código aberto
- **2016:** incorporado pelo © The Apache Software Foundation
- **2019:** anunciado como Top Level Project




run_this_last
run_this_last
run_this_last
run_this_last
run_this_last
Job do Zé
run_this_last
run_this_last
run_this_last
run_this_last
run_this_last


Airflow
DAGs
Data Profiling
Browse
Admin
Docs
About
2018-09-07 22:14:10 UTC
DAGs
Search:
DAG
Schedule
Owner
Recent Tasks
Last Run
DAG Runs
Links
On
example_bash_operator
0 0 * * *
airflow
2018-09-06 00:00
On
example_branch_dop_operator_v3
*/1 * * * *
airflow
2018-09-05 00:56
On
example_branch_operator
@daily
airflow
2018-09-06 00:00
On
example_xcom
@once
airflow
2018-09-05 00:00
On
latest_only
4:00:00
Airflow
2018-09-07 16:00
Showing 1 to 5 of 5 entries
«
<
1
>
»
Show Paused DAGs


Airflow
DAGs
Data Profiling
Browse
Admin
Docs
About
2018-09-07 22:29:47 UTC
On
DAG: example_bash_operator
schedule: 0 0 * * *
Graph View
Tree View
Task Duration
Task Tries
Landing Times
Gantt
Details
Code
Refresh
Delete
success
Base date:
2018-09-06 00:00:01
Number of runs:
25
Run:
scheduled__2018-09-06T00:00:00+00:00
Layout:
Left->Right
Go
Search for...
BashOperator
DummyOperator
success
running
failed
skipped
retry
queued
no status
runme_0
runme_1
runme_2
run_after_loop
also_run_this
run_this_last

O que não é?

- Não é uma plataforma de stream

- Não é uma plataforma para publicar um serviço que rodar em loop

Instalação (Simples)

```
# airflow needs a home, ~/airflow is the default,
# but you can lay foundation somewhere else if you prefer
# (optional)
export AIRFLOW_HOME=~/airflow

# install from pypi using pip
pip install apache-airflow

# initialize the database
airflow initdb

# start the web server, default port is 8080
airflow webserver -p 8080

# start the scheduler
airflow scheduler

# visit localhost:8080 in the browser and enable the example dag in the home page
```

Instalação (Docker)

GitHub - puckel/docker-airflow

Apresenta dois tipos de executores: Local e Celery

Permite sobreposição dos arquivos:

- airflow.cfg
- entrypoint.sh
- requirements.txt

Instalação (Docker)

Customizações do projeto:

- Rotação de logs
- Instalação de client para ssh
- Utilização de MySQL

Conceitos principais

- DAGs
- Operators
- Sensors (ExternalTaskSensor)
- Connections (Hooks)
- Variables

Conceitos principais

Componentes do airflow

- webserver
- scheduler
- worker
- broker
- backend
- logs (locais e remotos)

Exemplos

- Show me the code!

Código python de módulos externos da DAG

```
import os
import sys
sys.path.insert(0, os.path.abspath(os.path.dirname(__file__)))
```

- Código no início de todo módulo que implementa uma DAG
- A DAG passa a ser a referência para a importação de módulos locais

Utilização de DagBag

DagBag é um conceito de organização do airflow

- Um script que indica ao Airflow onde as DAGs estão publicadas
- Garante uma melhor organização
- Permite a utilização de scripts auxiliares locais

OBS: é necessário que a DAG seja declarada no módulo que ela é usada, chamando explicitamente sua importação

Spark-Submit por SSH

- PythonOperator
  - Cria um subprocesso
  - Chama um shell script
- Permite a execução de instruções antes ou depois da execução do job
  - Consulta aos logs em caso de erro (implementei hoje de manhã)
- Configuração dos jobs do ct-ddp por classes auxiliares e parametrizadas

Visualização dos logs

- Logs do webserver
  - Conexões
  - Implementação da DAG
- Logs do scheduler
  - Execuções das tasks
  - Problemas de concorrência entre tasks
- Logs da task
  - Output do script stdout e stderr da task
  - Configurado para apontar para o s3

Execução de Task manualmente

- Task já executada com falha (state: failed)
  - Selecionar a task e clear ou run
- Task já executada com sucesso (state: success)
  - Selecionar a task e clear
- Task não executada
  - Selecionar a task e alterar o estado para failed, e depois clear no estado
  - Selecionar a task e run

OBS:

- Alterar o estado de uma task para clear, reativa a execução da DAG, executando qualquer Task
- Tomar cuidado com as opções de Downstream

Criação de módulos auxiliares

- Possibilita o reuso de código e manutenção
- Esses módulos devem ser publicados junto ao DagBag para uso global

Usos:

- Definição de Operadores padrões
- Queries a banco de dados
- Fluxos de tasks padrões (start >> copy_ssh_key >> customer >>)

Obrigado!