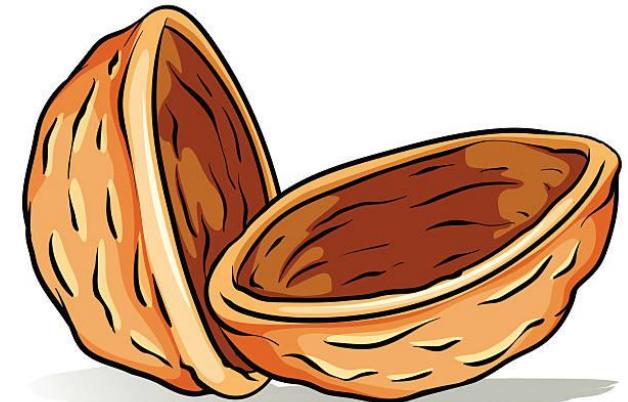




Cron em esteróides: **Apache Airflow** **(Brain-on** **Hands-on)**

Olá, nózes!



Nitai Bezerra

nitai.silva

@planejamento.gov.br

t. 2020-1551



Vitor Bellini

vitor.bellini

@planejamento.gov.br

t. 2020-1504



ecodados.economia.gov.br

Ecodados - Fórum dos cien + https://ecodados.economia.gov.br 133% 🔍 🌐 ⚡

Ecodados

Para facilitar o lançamento do seu novo site, você está no modo de bootstrap. Todos os novos usuários receberão o nível de confiança 1 e terão os e-mails diários de resumo ativados. Isso será desativado automaticamente quando 100 usuários se registrarem.

todas as categorias ▶ Recente Melhores Categorias + Novo Tópico

Tópico Respostas Visualizações Atividade

Scheduling Notebooks at Netflix medium.com ■ Artigos e Publicações jupyter, airflow, notebooks	 1 7 4d
Voltamos! ou: renovação do certificado digital ■ Equipe downtime, lets-encrypt	 3 6 6d
Base de Acórdãos TCU ■ Projetos e Experimentos	 2 11 6d
Voltamos ao ar! ■ Meta downtime, lets-encrypt	 0 14 7d
Precisamos realmente de tantos Cientistas de Dados? ■ Artigos e Publicações airflow	 0 32 18d
Surfando na crista da onda de visualização de dados (Superset e Metabase) ■ Artigos e Publicações visualização, dashboard	 7 65 19d
On the Philosophical Foundations of Conceptual Models ■ Artigos e Publicações modelagem-conceptual	 1 19 21d

E aí!? Como vai ser?

1. Quebrando o gelo gelado
2. Ao infinito e além: filosofia e arquitetura Airflow
3. Airflow UI, seu lindo!
4. DAG: que danado é isso!?
5. Baleia intro (Docker)
6. Botar pra rodar
7. SELECT test FROM oficina
8. [break] Ufa, DESCANSO (20 min.)

E aí!? Como vai ser?

8. [break] Ufa, DESCANSO
9. Hello World: Seu primeiro robô (ou não)
10. Baby Steps: Implementação de uma DAG pra chamar de sua
11. Exemplos DAGs ME
12. Dicas e Truques (quem souber me conta)
13. Feedback



1.

Quebrando o
gelo gelado

Origami + Enigma



Washington Oliveira

Porque estamos aqui!?

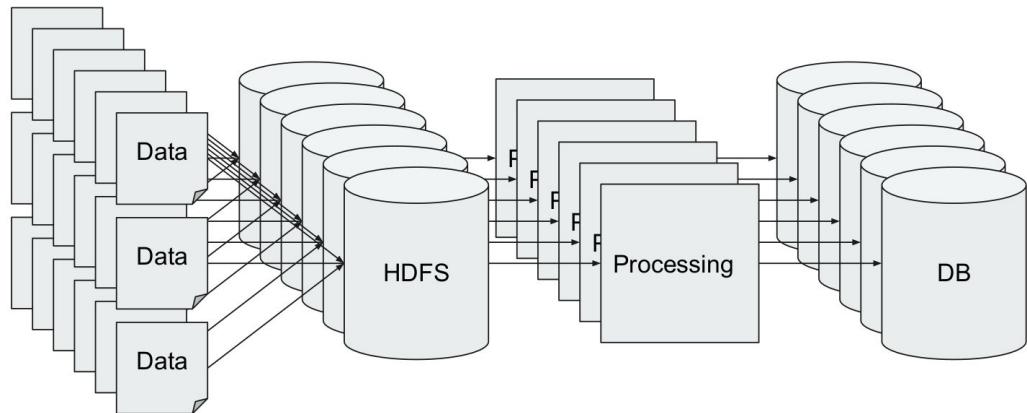
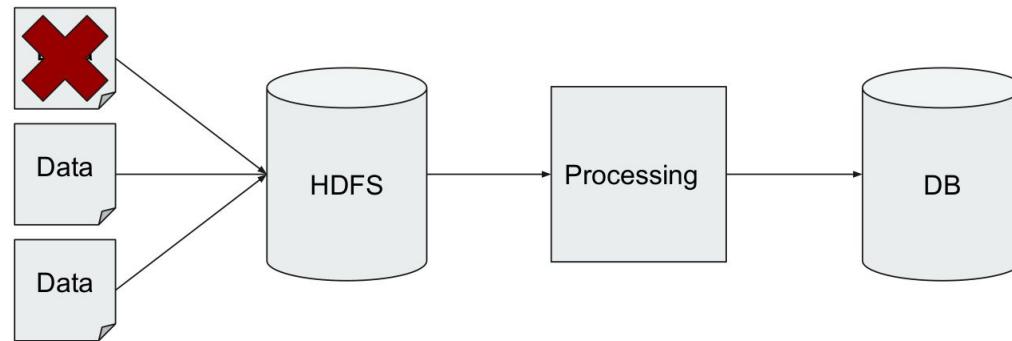
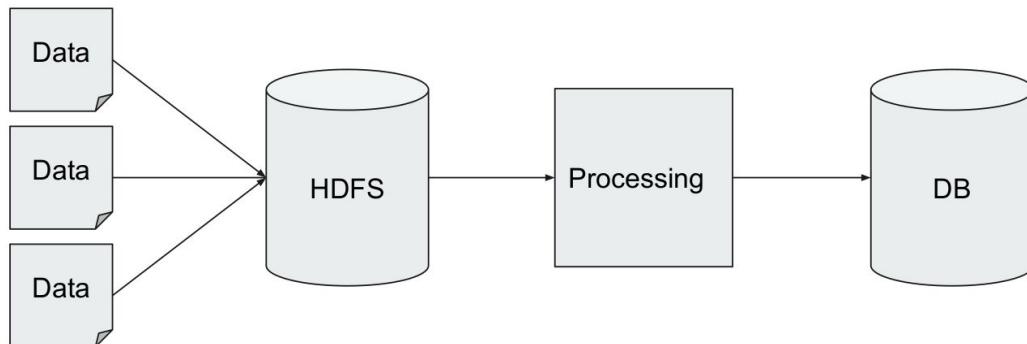




2.

Ao infinito e
além: filosofia
e arquitetura
Airflow

Porque usar?



Para isso e mais...

- Substituir rotinas Cron
- Tolerancia à falhas
- Regras de dependência
- Python!!!
- Versionamento dos códigos

Para isso e mais...

- Tratamento de erros
- Relatórios e alertas em falhas
- Monitoramento de pipelines
- UI bonita pacas!

O que é!? Já aprendeu a voar?

Apache Airflow is a way to programmatically author, schedule, and monitor data pipelines





Componentes

○ **WebServer**

- UI para monitoramento dos jobs e outras coisas

○ **Scheduler**

- Server pra agendar os jobs

○ **Executor**

- É o chefe intermediário. O **Scheduler** diz pra ele o que fazer e ele delega para o **Worker**

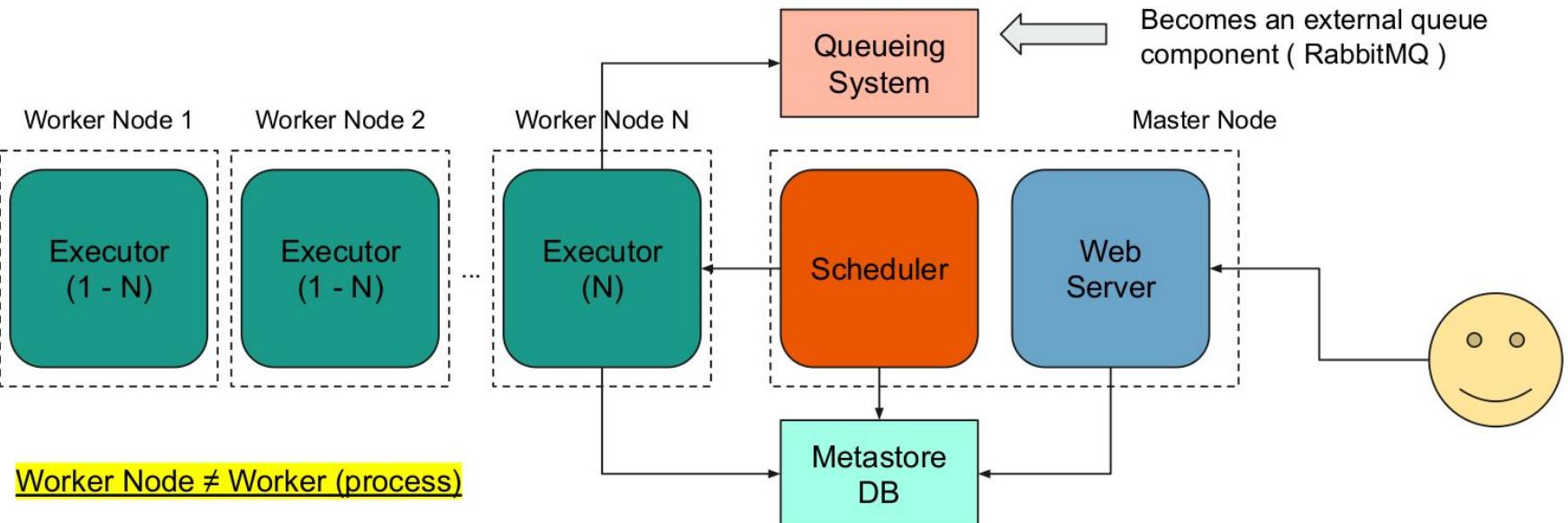
○ **Worker**

- Executa a tarefa

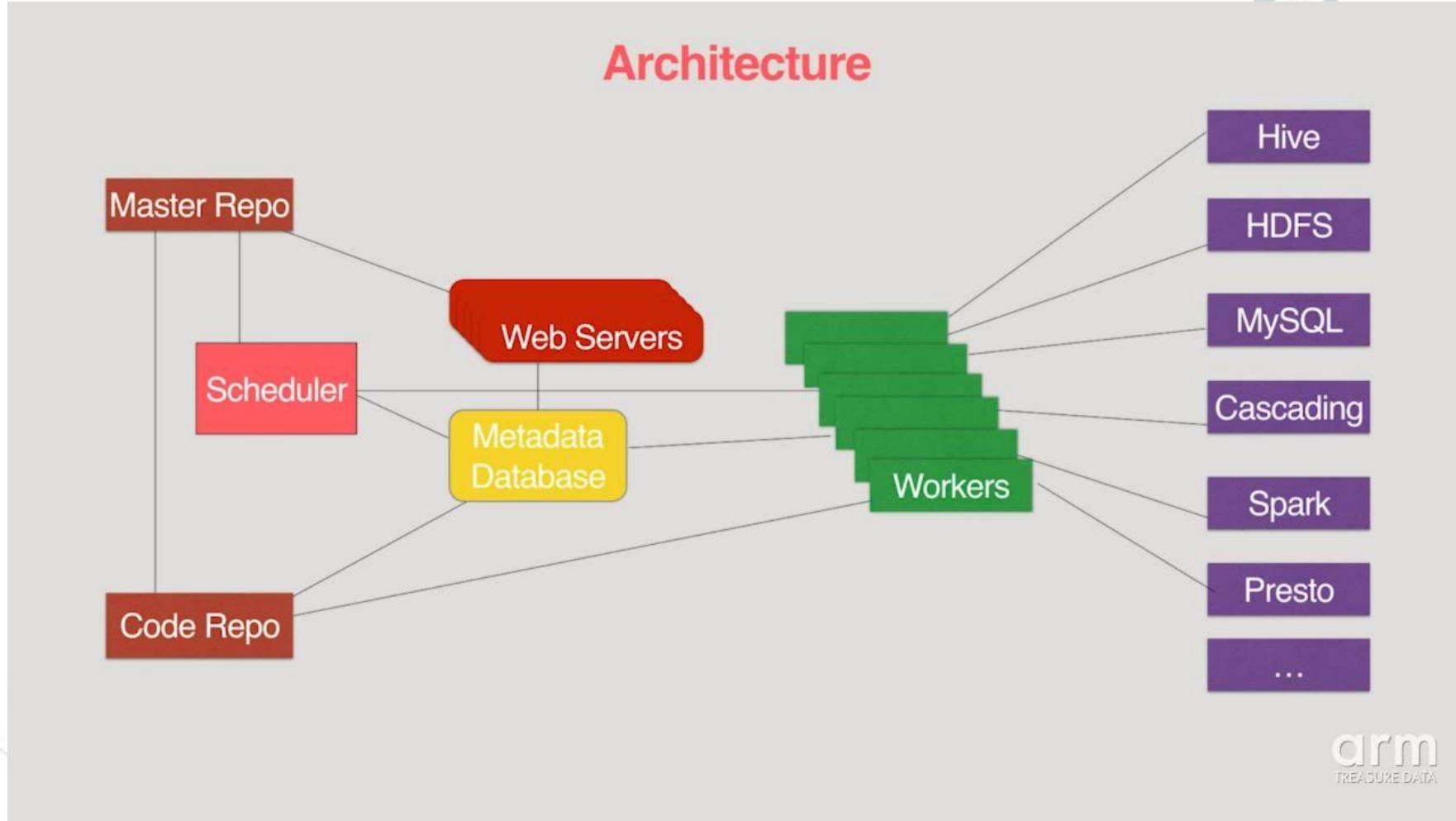
○ **Metadatabase**

- Armazenamento das informações sobre as execuções das DAGs

Visão da Arquitetura



Visão da Arquitetura



Conceitos Chave



- ◎ **DAG**
 - Grafo que representa o **pipeline de dados**
- ◎ **Operator**
 - Descreve uma tarefa única na **DAG**
- ◎ **Task**
 - A instância do **operador**
- ◎ **Task Instance**
 - Representa uma execução específica de uma **task**
 - **= DAG + TASK + POINT IN TIME**
- ◎ **Workflow**
 - Combinação de todos os acima

O que o Airflow não é



Solução de stream de dados

- Não está no escopo do Apache Spark ou Storm
- Construído inicialmente para executar batch jobs agendados

3.

Airflow UI,
seu lindo!



DAGs View

Lista dos DAGs no seu ambiente e um conjunto de atalhos para páginas úteis. Você pode ver exatamente quantas tarefas foram bem-sucedidas, falharam ou estão em execução no momento.

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
	example_bash_operator		airflow		2018-09-06 00:00		
	example_branch_dop_operator_v3		airflow		2018-09-05 00:56		
	example_branch_operator		airflow		2018-09-06 00:00		
	example_xcom	@once icon"/>	airflow		2018-09-05 00:00		
	latest_only		Airflow		2018-09-07 16:00		

Showing 1 to 5 of 5 entries

< 1 > >>

Show Paused DAGs

Tree View

Uma representação em árvore do DAG que se estende através do tempo. Se um pipeline estiver atrasado, você poderá ver rapidamente onde estão as diferentes etapas e identificar as que estão bloqueando.

Airflow DAGs Data Profiling Browse Admin Docs About 2018-09-07 22:15:40 UTC ⏪

On DAG: example_branch_dop_operator_v3 schedule: */1 * * * *

Graph View Tree View Task Duration Task Tries Landing Times Gantt Details Code Refresh Delete

Base date: 2018-09-05 01:04:00 Number of runs: 25 Go

BranchPythonOperator DummyOperator

success running failed skipped retry queued no status

The screenshot shows the Airflow web interface with the 'Tree View' tab selected. At the top, there's a navigation bar with links for Airflow, DAGs, Data Profiling, Browse, Admin, Docs, and About, along with a timestamp and a refresh button. Below the navigation is a header for the DAG 'example_branch_dop_operator_v3' with a scheduled cron expression of '*/1 * * * *'. A toolbar below the header includes Graph View, Tree View (selected), Task Duration, Task Tries, Landing Times, Gantt, Details, Code, Refresh, and Delete. The main content area has sections for 'Base date' (set to 2018-09-05 01:04:00) and 'Number of runs' (set to 25). A 'Go' button is next to the run count. Below these are two legend sections: one for task types (BranchPythonOperator, DummyOperator) and one for task states (success, running, failed, skipped, retry, queued, no status). On the left, a tree view diagram shows the DAG structure with nodes like '[DAG]', 'oper_1', 'condition', 'oper_2', and another 'condition'. To the right, a timeline chart displays task execution from 05:45 to 06 PM. The timeline consists of a grid of colored dots representing task runs. The legend indicates that green dots represent 'success', while other colors (pink, yellow, red) represent different failure or skip states. The timeline shows multiple runs for each task, with some tasks having more successful runs than others.

Graph View

A visualização do gráfico é talvez a mais abrangente.
Visualize as dependências do seu DAG e seu status atual
para uma execução específica.

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

```
graph TD; runme_0 --> run_after_loop; runme_1 --> run_after_loop; runme_2 --> also_run_this; runme_1 --> also_run_this; runme_1 --> run_this_last; runme_2 --> run_this_last;
```

Variable View

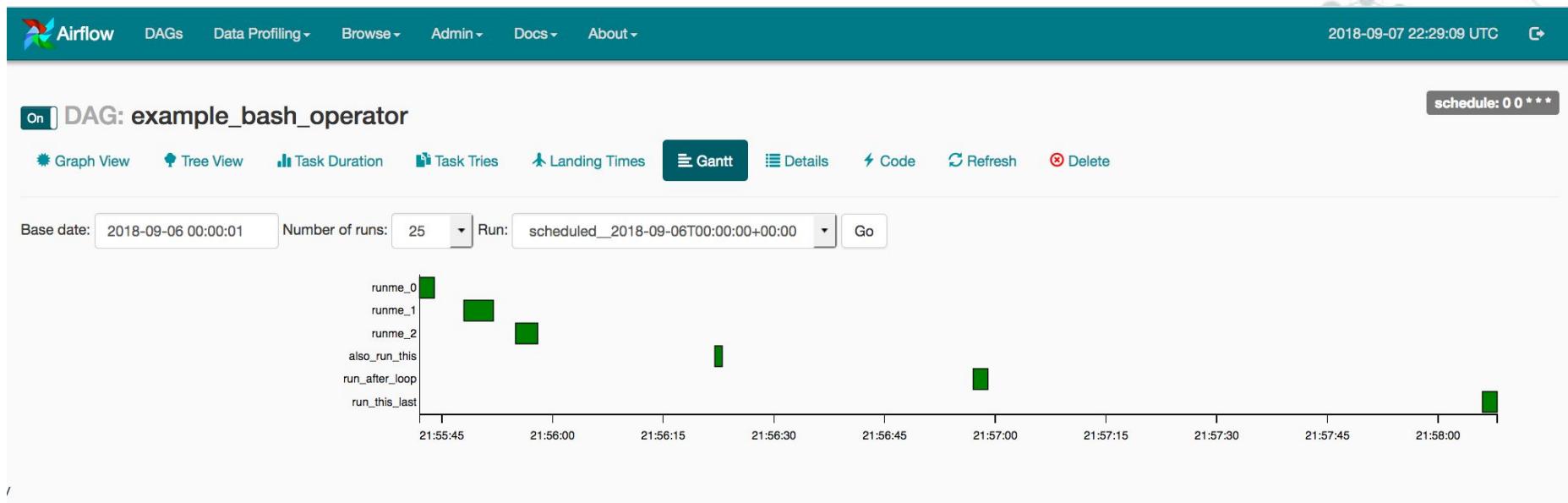
A visualização de variável permite listar, criar, editar ou excluir o par de valores-chave de uma variável usada durante os trabalhos. O valor de uma variável ficará oculto se a chave contiver alguma palavra em ('senha', 'segredo', 'senha', 'autorização', 'api_key', 'apikey', 'access_token') por padrão, mas poderá ser configurada para mostrar em texto não criptografado.

The screenshot shows the Airflow Variables page. At the top, there is a navigation bar with links for Airflow, DAGs, Data Profiling, Browse, Admin, and Docs. Below the navigation bar, the page title is "Variables". There are several buttons at the top of the list: "List (9)", "Create", "Add Filter", "With selected", and a search input field. The main content is a table with two columns: "Key" and "Val". Each row contains a checkbox, edit/pencil icons, and delete/trash icons. The "Val" column displays values as asterisks (*****) for most entries, except for "not_so_hidden" which shows "test value".

	Key	Val
<input type="checkbox"/>	secret_password	*****
<input type="checkbox"/>	not_so_hidden	test value
<input type="checkbox"/>	secret	*****
<input type="checkbox"/>	password	*****
<input type="checkbox"/>	passwd	*****
<input type="checkbox"/>	api_key	*****
<input type="checkbox"/>	apikey	*****
<input type="checkbox"/>	authorization	*****
<input type="checkbox"/>	access_token	*****

Gantt Chart

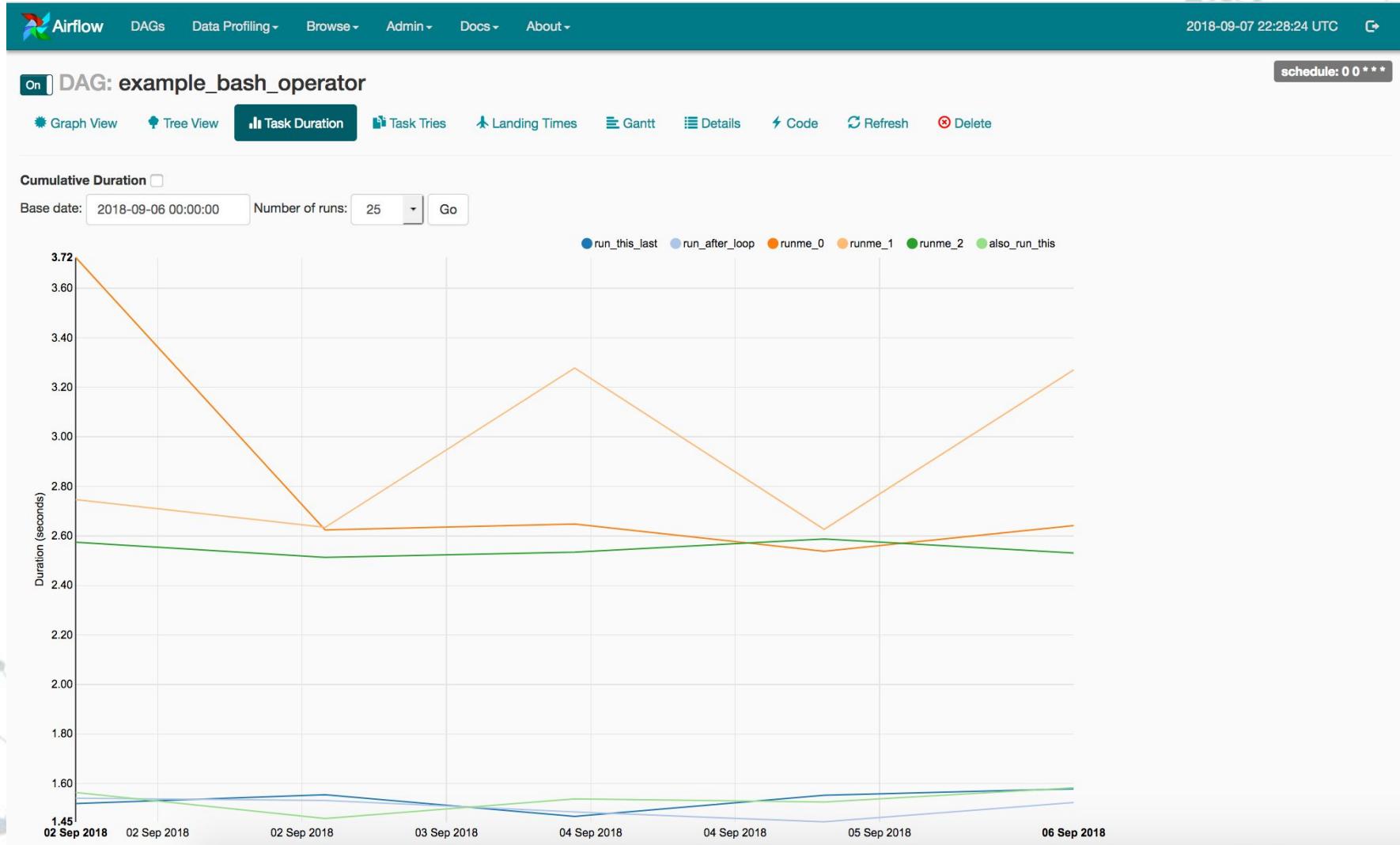
O gráfico de Gantt permite analisar a duração e a sobreposição de tarefas. Você pode identificar rapidamente gargalos e onde a maior parte do tempo é gasta para execuções específicas do DAG.



fonte: <https://airflow.apache.org/ui.html>

Task Duration

A duração de suas diferentes tarefas nas N corridas anteriores. Essa visualização permite encontrar discrepâncias e entender rapidamente onde o tempo é gasto no seu DAG em várias execuções.



Code View

Transparência é tudo. Enquanto o código do seu pipeline está no controle de origem, esta é uma maneira rápida de acessar o código que gera o DAG e fornece ainda mais contexto.

The screenshot shows the Apache Airflow web interface with the following details:

- Header:** Airflow, DAGs, Data Profiling, Browse, Admin, Docs, About.
- Breadcrumbs:** On | DAG: example_bash_operator
- Buttons:** Graph View, Tree View, Task Duration, Task Tries, Landing Times, Gantt, Details, **Code** (highlighted), Refresh, Delete.
- Schedule:** 0 0 ***
- Code Content:**

```
1 # -*- coding: utf-8 -*-
2 #
3 # Licensed to the Apache Software Foundation (ASF) under one
4 # or more contributor license agreements. See the NOTICE file
5 # distributed with this work for additional information
6 # regarding copyright ownership. The ASF licenses this file
7 # to you under the Apache License, Version 2.0 (the
8 # "License"); you may not use this file except in compliance
9 # with the License. You may obtain a copy of the License at
10 #
11 #     http://www.apache.org/licenses/LICENSE-2.0
12 #
13 # Unless required by applicable law or agreed to in writing,
14 # software distributed under the License is distributed on an
15 # "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
16 # KIND, either express or implied. See the License for the
17 # specific language governing permissions and limitations
18 # under the License.
19
20 import airflow
21 from builtins import range
22 from airflow.operators.bash_operator import BashOperator
23 from airflow.operators.dummy_operator import DummyOperator
24 from airflow.models import DAG
25 from datetime import timedelta
26
27
28 args = {
29     'owner': 'airflow',
30     'start_date': airflow.utils.dates.days_ago(2)
31 }
32
33 dag = DAG(
34     dag_id='example_bash_operator', default_args=args,
35     schedule_interval='@ * * * *',
36     dagrun_timeout=timedelta(minutes=60))
37
38 cmd = 'ls -l'
39 run_this_last = DummyOperator(task_id='run_this_last', dag=dag)
40
41 # [START howto_operator_bash]
42 run_this = BashOperator(
43     task_id='run_after_loop', bash_command='echo 1', dag=dag)
44 # [END howto_operator_bash]
45 run_this.set_downstream(run_this_last)
```

fonte: <https://airflow.apache.org/ui.html>

Task Instance Context Menu

Nas páginas vistas acima (exibição em árvore, exibição de gráfico, gantt,...), sempre é possível clicar em uma instância de tarefa e acessar esse menu de contexto avançado que pode levá-lo a metadados mais detalhados e executar algumas ações.

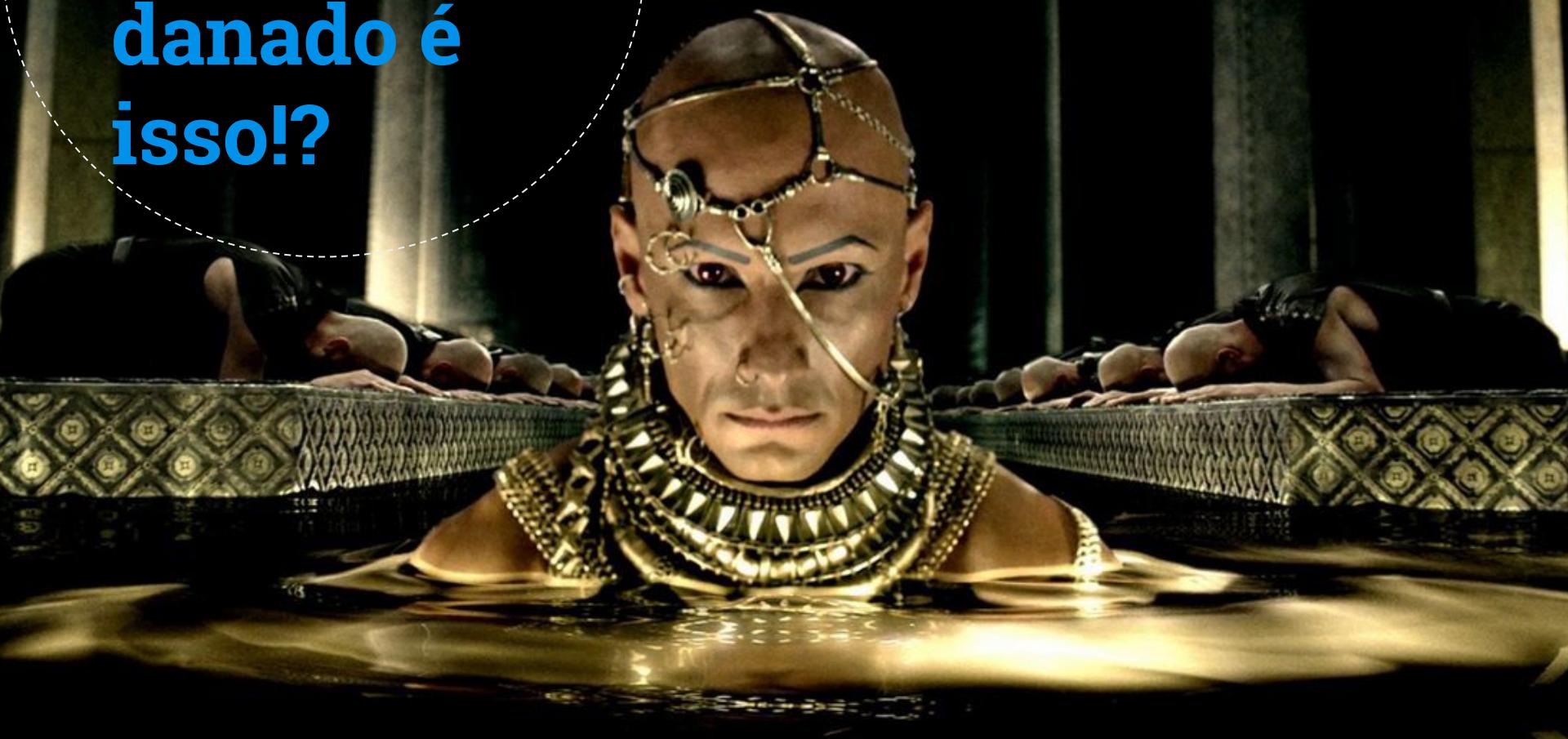
The screenshot shows the Apache Airflow web interface for a DAG named 'example_bash_operator'. The DAG has three tasks: 'runme_0', 'runme_1', and 'runme_2'. 'runme_1' has two downstream tasks: 'run_after_loop' and 'also_run_this'. 'runme_2' also has 'run_after_loop' as a downstream task. A context menu is open over the 'runme_1' task, specifically over its 'run_after_loop' instance. The menu is titled 'runme_1 run_after_loop on 2018-09-08T00:00:00+00:00'. It contains several tabs and buttons:

- Task Instance Details (selected)
- Rendered
- Task Instances
- View Log
- Run
- Ignore All Deps
- Ignore Task State
- Ignore Task Deps
- Clear
- Past
- Future
- Upstream
- Downstream
- Recursive
- Mark Failed
- Past
- Future
- Upstream
- Downstream
- Mark Success
- Past
- Future
- Upstream
- Downstream

A 'Close' button is at the bottom right of the menu. The background shows the DAG's structure and some status indicators like 'schedule: 0 0 * * *' and 'success running failed skipped retry queued no status'.

4.

DAG: que
danado é
isso!?

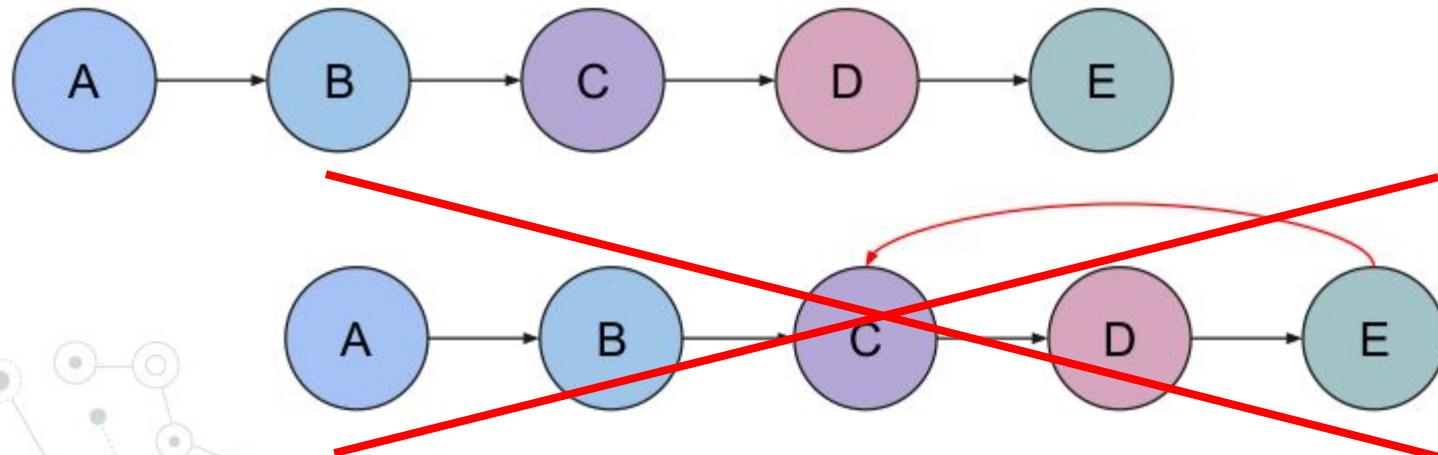


Definição de DAG

- ◎ Conceitualmente uma DAG (Grafo Acíclico Direcionado) é um **grafo direcionado** finito que não possui ciclos (loops).
- ◎ No Airflow, uma DAG representa um conjunto de **tarefas** a serem executadas, organizadas de maneira a representar suas dependências e relacionamentos.
 - Cada Nó é uma tarefa
 - Cada Aresta é uma dependência

Um exemplo fictício simples

- Nó A faz download de arquivo de dados
- Nó B envia os dados para processamento
- Nó C monitora e aguarda conclusão do processamento
- Nó D atualiza indicador no BD
- Nó E envia notificação de conclusão via Telegram



O que são Operadores?

- ◎ Um operador descreve uma única Tarefa em um Workflow
- ◎ Geralmente os operadores são atômicos
 - Não compartilham dados com outro operador
- ◎ São reexecutados em caso de falhas
- ◎ A DAG garante executar os operadores na ordem correta

Operadores nativos do Airflow

- BashOperator
 - Executa um comando Bash
- PythonOperator
 - Chama uma função python qualquer
- EmailOperator
 - Envia um email
- MySqlOperator, SqliteOperator,
PostgreOperator...
 - Executa um comando SQL

Tipos de Operadores

Todos Operadores herdam de *BaseOperator*.

Existem 3 tipos de operadores:

- ◎ Operadores de **Ação**

- (BashOperator, PythonOperator, EmailOperator...)

- ◎ Operadores de **Transferência**

- (MongoToS3Operator, MySqlToHiveTransfer, SFTPOperator...)

- ◎ Operadores **Sensores**

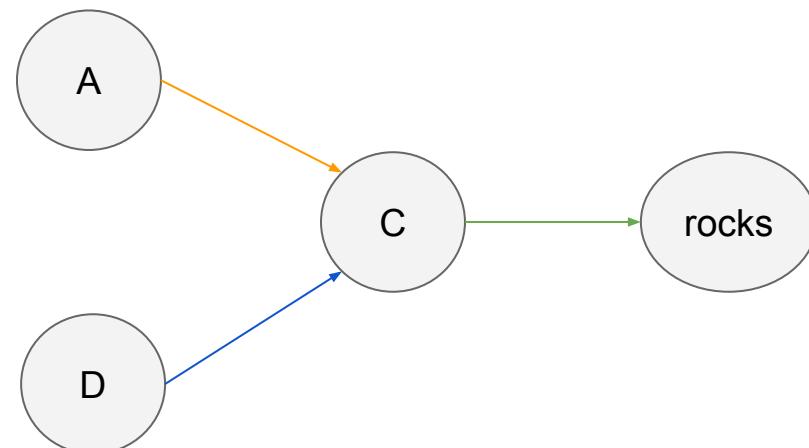
- (SqlSensor, HttpSensor, HdfsSensor...)

Operadores pra todo gosto...

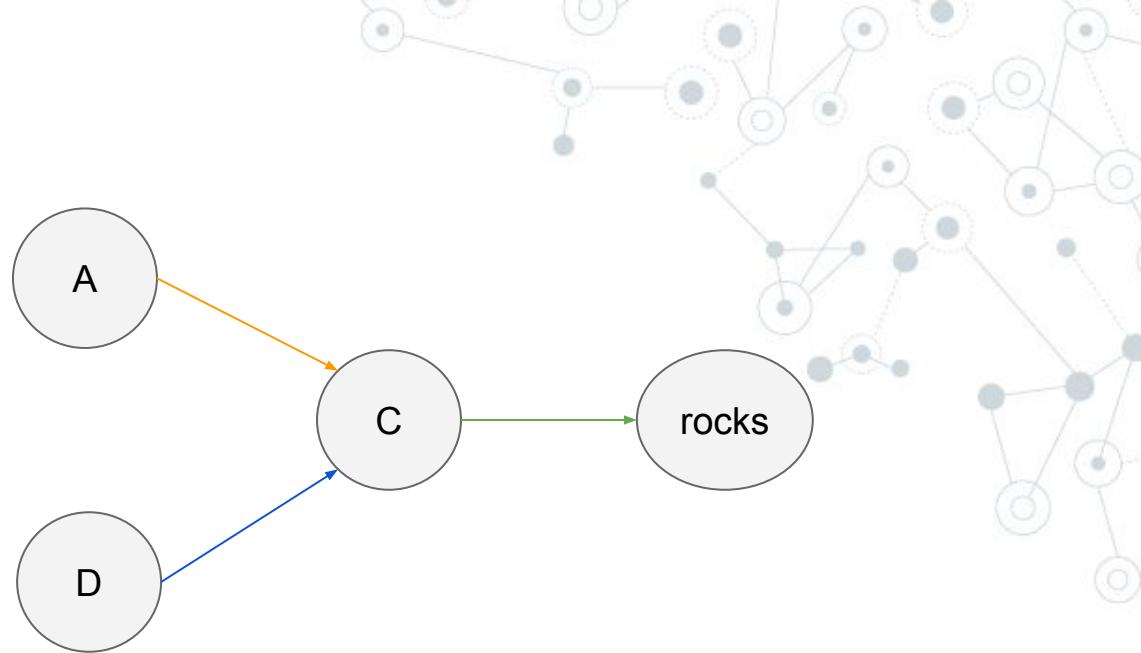
- `airflow.operators.bash_operator`
- `airflow.operators.branch_operator`
- `airflow.operators.check_operator`
- `airflow.operators.dagrun_operator`
- `airflow.operators.docker_operator`
- `airflow.operators.druid_check_operator`
- `airflow.operators.dummy_operator`
- `airflow.operators.email_operator`
- `airflow.operators.generic_transfer`
- `airflow.operators.hive_operator`
- `airflow.operators.hive_stats_operator`
- `airflow.operators.hive_to_druid`
- `airflow.operators.hive_to_mysql`
- `airflow.operators.hive_to_samba_operator`
- `airflow.operators.http_operator`
- `airflow.operators.jdbc_operator`
- `airflow.operators.latest_only_operator`
- `airflow.operators.mssql_operator`
- `airflow.operators.mssql_to_hive`
- `airflow.operators.mysql_operator`
- `airflow.operators.mysql_to_hive`
- `airflow.operators.oracle_operator`
- `airflow.operators.pig_operator`
- `airflow.operators.postgres_operator`
- `airflow.operators.presto_check_operator`
- `airflow.operators.presto_to_mysql`
- `airflow.operators.python_operator`
- `airflow.operators.redshift_to_s3_operator`
- `airflow.operators.s3_file_transform_operator`
- `airflow.operators.s3_to_hive_operator`
- `airflow.operators.s3_to_redshift_operator`
- `airflow.operators.sensors`
- `airflow.operators.slack_operator`
- `airflow.operators.sqlite_operator`
- `airflow.operators.subdag_operator`
- `airflow.contrib.operators.bigquery_to_bigquery`
- `airflow.contrib.operators.bigquery_to_gcs`
- `airflow.contrib.operators.cassandra_to_gcs`
- `airflow.contrib.operators.databricks_operator`
- `airflow.contrib.operators.dataflow_operator`
- `airflow.contrib.operators.dataproc_operator`
- `airflow.contrib.operators.datastore_export_operator`
- `airflow.contrib.operators.datastore_import_operator`
- `airflow.contrib.operators.dingding_operator`
- `airflow.contrib.operators.discord_webhook_operator`
- `airflow.contrib.operators.druid_operator`
- `airflow.contrib.operators.ecs_operator`
- `airflow.contrib.operators.emr_add_steps_operator`
- `airflow.contrib.operators.emr_create_job_flow_operator`
- `airflow.contrib.operators.emr_terminate_job_flow_operator`
- `airflow.contrib.operators.file_to_gcs`
- `airflow.contrib.operators.file_to_wasb`
- `airflow.contrib.operators.gcp_bigtable_operator`
- `airflow.contrib.operators.gcp_cloud_build_operator`
- `airflow.contrib.operators.gcp_compute_operator`
- `airflow.contrib.operators.gcp_container_operator`
- `airflow.contrib.operators.gcp_dlp_operator`
- `airflow.contrib.operators.gcp_function_operator`
- `airflow.contrib.operators.gcp_natural_language_operator`

Compondo Tarefas

- Nesta DAG, os nós são A, D, C e rocks
- As arestas são as setas coloridas
- Cada seta representa uma dependência
 - **C** depende de **A** e **D**
 - **rocks** depende de **C**



Exemplo:



As dependências podem ser expressas de 4 formas:

- Ⓐ `A.set_downstream(C); D.set_downstream(C);
D.set_downstream(rocks)`
- Ⓑ `rocks.set_upstream(C); C.set_upstream(A); C.set_upstream(D)`
- Ⓒ `rocks << C; C << A; C << D`
- Ⓓ `A >> C; D >> C >> rocks`

Parâmetros da DAG

○ `start_date`

- A primeira data para a qual você deseja que os dados sejam produzidos pela DAG (pode ser definido no passado)

○ `end_date`

- A data em que sua DAG deve parar de ser executado (none por padrão)

○ `retries`

- O número máximo de tentativas antes da Task falhar

○ `retry_delay`

- O tempo de espera até a tentativa seguinte

○ `schedule_interval`

O intervalo em que o Scheduler acionará sua DAG

schedule_interval

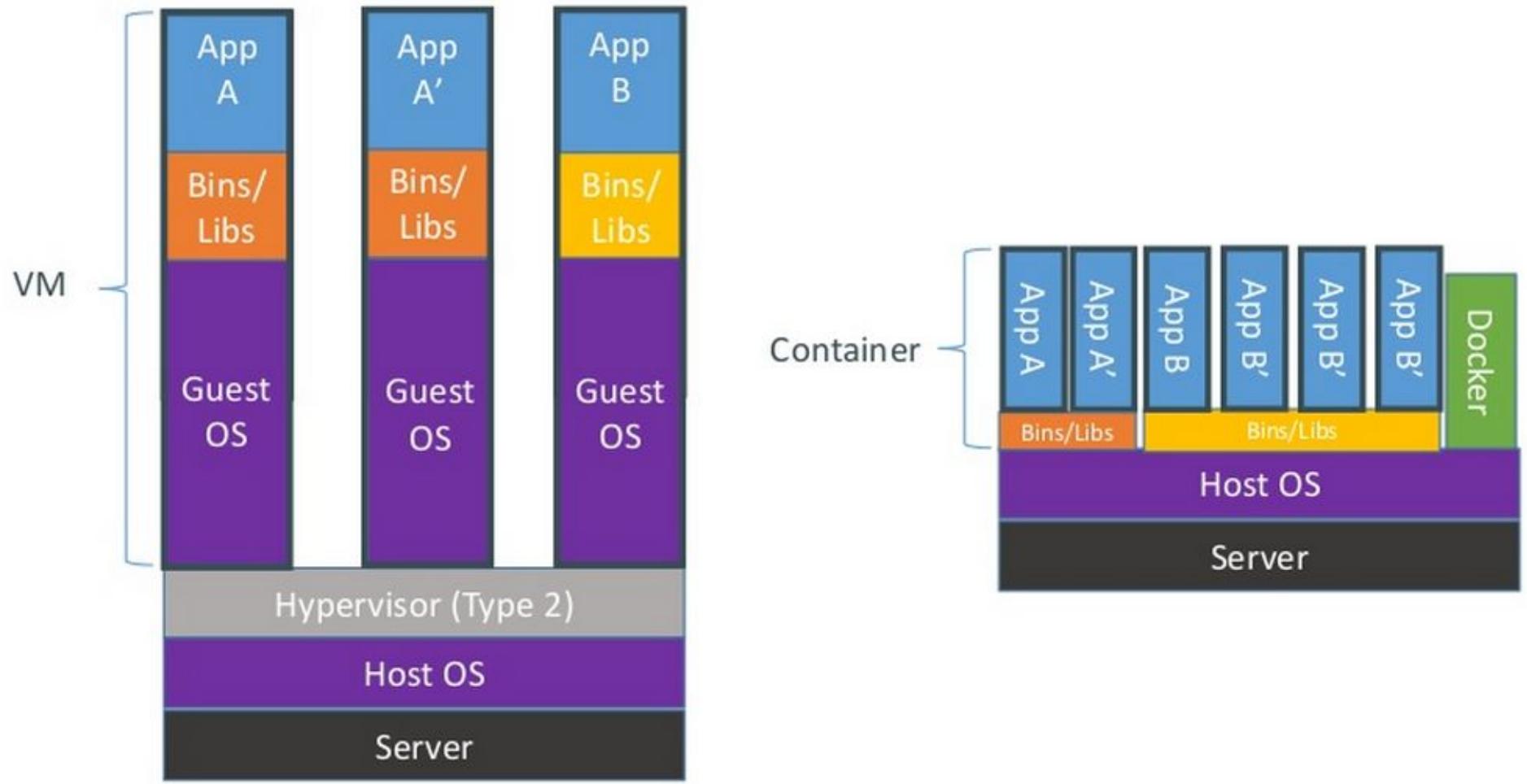
- Indica em qual intervalo o Scheduler deve rodar a sua DAG
- Aceita expressões CRON, datetime.timedelta ou presets.

Predefinidos	Resultado	Cron
None	Não agenda. Acionado manualmente	
@once	Agenda uma e somente uma vez	
@hourly	Roda uma vez por hora no início da hora	0 * * * *
@daily	Roda uma vez por dia à meia-noite	0 0 * * *
@weekly	Roda uma vez por semana à meia-noite no domingo cedo	0 0 * * 0
@monthly	Roda uma vez por mês à meia-noite do primeiro dia do mês	0 0 1 * *
@yearly	Roda uma vez por ano à meia-noite de 1º de Janeiro	0 0 1 1 *



5. **Baleia intro** (Docker)

Containers vs VMs

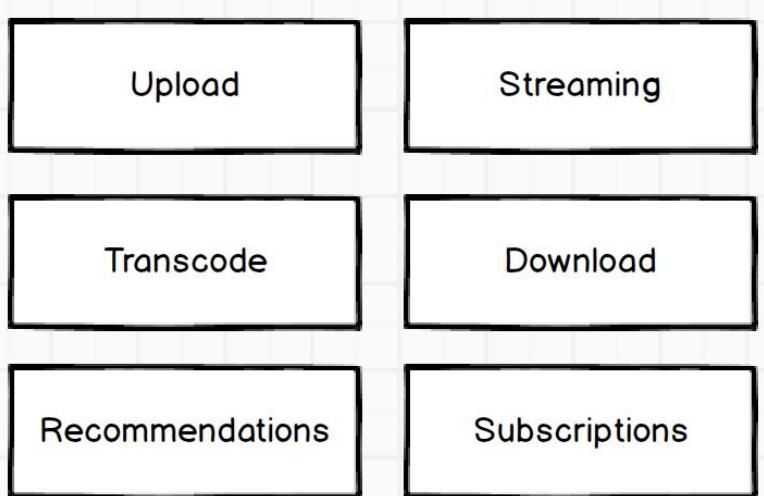


Monolítico vs Microserviços

Monolith



Microservices



Venha ser feliz com Docker!

- Isolamento
- Execução leve
- Simplicidade
- Workflow DevOps
- Comunidade



A blurred image of a police car with its lights on, driving at night. The car is dark-colored with blue and red emergency lights visible. The background shows a city skyline with lit buildings.

6.

**Botar pra
rodar**

Pool

Levanta a mão quem não tem o docker instalado na máquina

```
if 🙋‍♂️ and SO == "Linux": #eeeeee
```

<https://docs.docker.com/install/>

```
elif 🙋‍♂️ and SO == "Windows": #uuuu
```

<https://docs.docker.com/docker-for-windows/install/>

```
else:
```

```
    print("Suave na nave")
```

Github

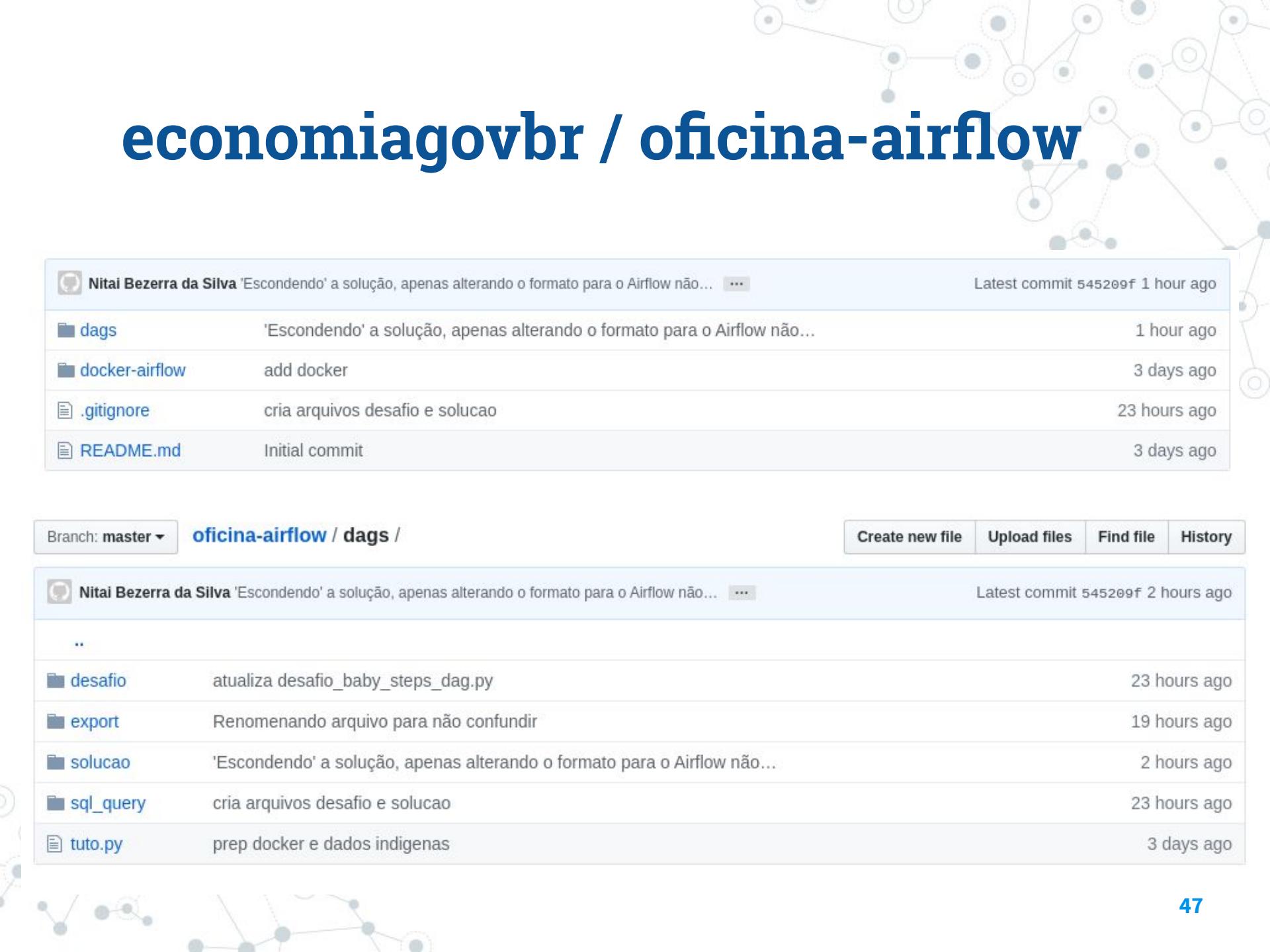
No terminal:

```
$ git clone
```

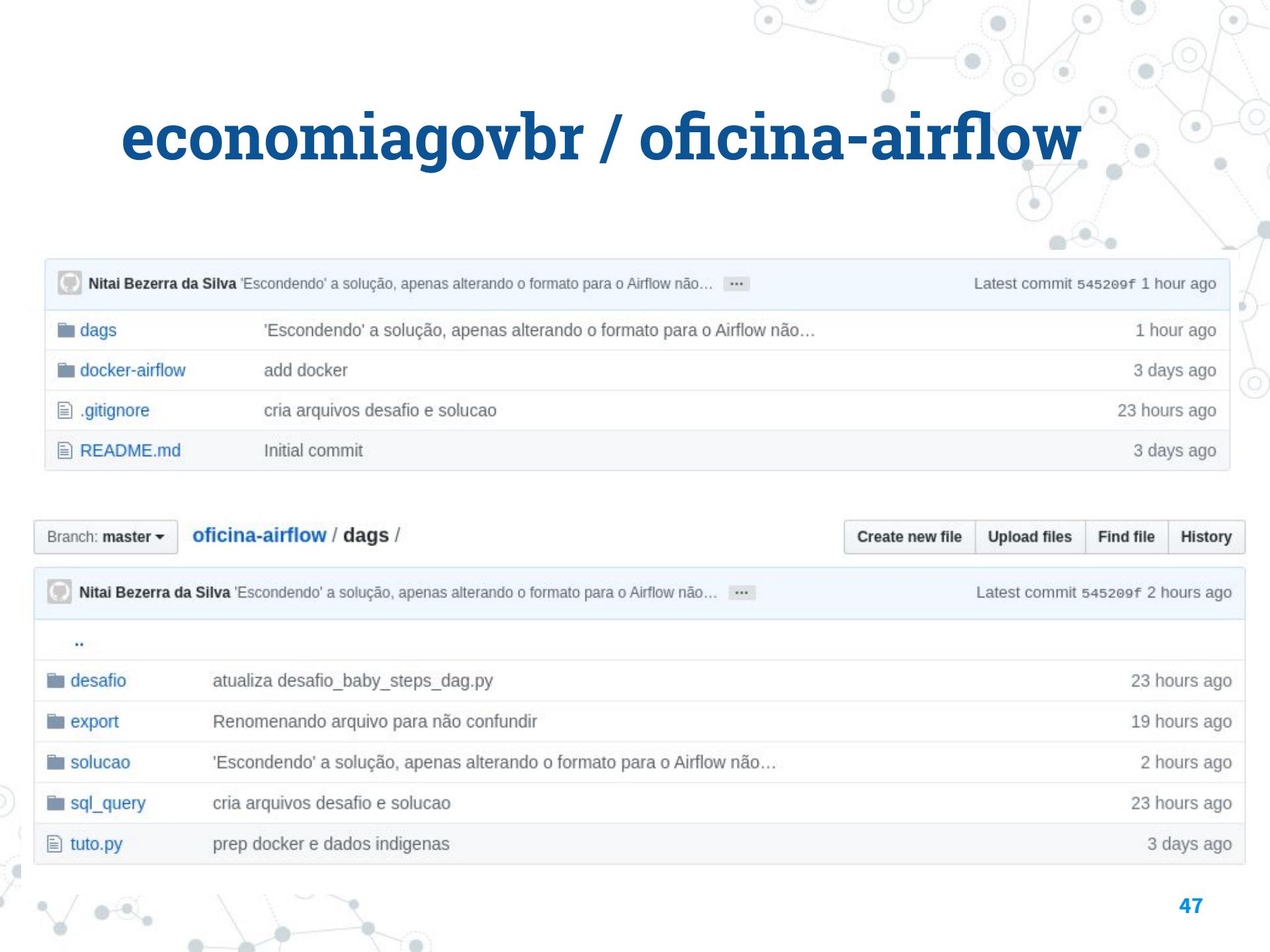
```
https://github.com/economiagovbr/oficina-airflow.git
```

(também conhecido como download com sincronização)

economiagovbr / oficina-airflow



 Nitai Bezerra da Silva	'Escondendo' a solução, apenas alterando o formato para o Airflow não...	...	Latest commit 545209f 1 hour ago
 dags	'Escondendo' a solução, apenas alterando o formato para o Airflow não...	1 hour ago	
 docker-airflow	add docker	3 days ago	
 .gitignore	cria arquivos desafio e solucao	23 hours ago	
 README.md	Initial commit	3 days ago	



Branch: master	oficina-airflow / dags /	Create new file	Upload files	Find file	History
 Nitai Bezerra da Silva	'Escondendo' a solução, apenas alterando o formato para o Airflow não...	...	Latest commit 545209f 2 hours ago		
	..				
 desafio	atualiza desafio_baby_steps_dag.py	23 hours ago			
 export	Renomenando arquivo para não confundir	19 hours ago			
 solucao	'Escondendo' a solução, apenas alterando o formato para o Airflow não...	2 hours ago			
 sql_query	cria arquivos desafio e solucao	23 hours ago			
 tuto.py	prep docker e dados indigenas	3 days ago			

Github

No terminal:

```
$ cd oficina-airflow/docker-airflow  
$ docker-compose up
```

(também conhecido como:
magia)



Se funcionou sua tela está assim

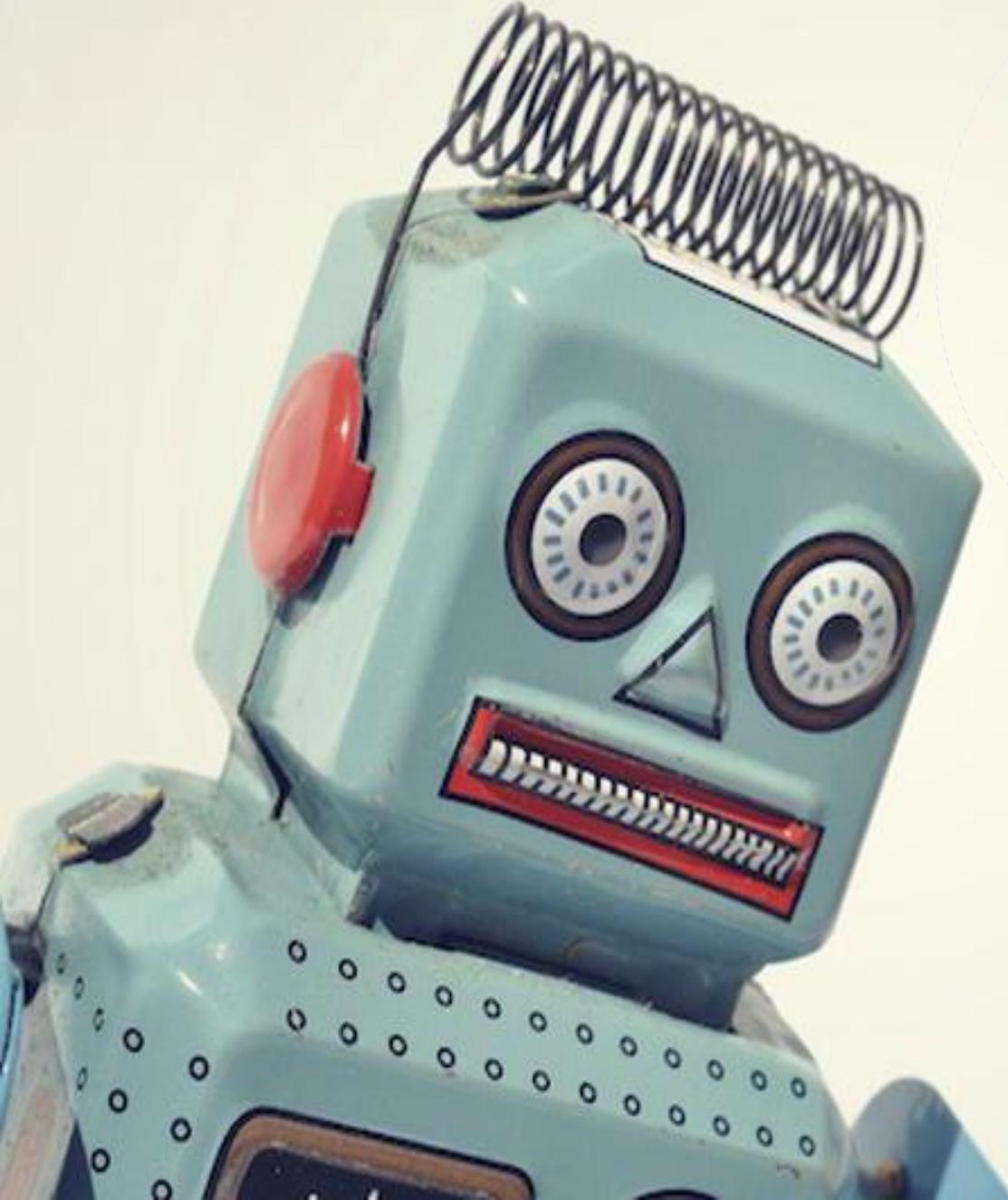
```
● ○ ● docker-compose
webserver_1 | 172.20.0.1 - - [15/Sep/2019:18:13:30 +0000] "GET /admin/ HTTP/1.1" 200 27710 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100 Safari/537.36"
webserver_1 | 172.20.0.1 - - [15/Sep/2019:18:13:30 +0000] "GET /admin/airflow/blocked HTTP/1.1" 200 2 "http://localhost:8080/admin/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100 Safari/537.36"
webserver_1 | 172.20.0.1 - - [15/Sep/2019:18:13:30 +0000] "GET /admin/airflow/dag_stats HTTP/1.1" 200 442 "http://localhost:8080/admin/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100 Safari/537.36"
webserver_1 | 172.20.0.1 - - [15/Sep/2019:18:13:30 +0000] "GET /admin/airflow/task_stats HTTP/1.1" 200 1445 "http://localhost:8080/admin/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100 Safari/537.36"
webserver_1 | 172.20.0.1 - - [15/Sep/2019:18:13:36 +0000] "GET /admin/airflow/tree?dag_id=tutorial HTTP/1.1" 200 9298 "http://localhost:8080/admin/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100 Safari/537.36"
webserver_1 | 172.20.0.1 - - [15/Sep/2019:18:13:36 +0000] "GET /admin/admin/vendor/select2/select2.css?v=3.5.2 HTTP/1.1" 200 0 "http://localhost:8080/admin/airflow/tree?dag_id=tutorial" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100 Safari/537.36"
webserver_1 | 172.20.0.1 - - [15/Sep/2019:18:13:36 +0000] "GET /admin/admin/vendor/select2/select2-bootstrap3.css?v=1.4.6 HTTP/1.1" 200 0 "http://localhost:8080/admin/airflow/tree?dag_id=tutorial" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100 Safari/537.36"
webserver_1 | 172.20.0.1 - - [15/Sep/2019:18:13:36 +0000] "GET /admin/admin/vendor/bootstrap-daterangepicker-bs3.css?v=1.3.22 HTTP/1.1" 200 0 "http://localhost:8080/admin/airflow/tree?dag_id=tutorial" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100 Safari/537.36"
webserver_1 | 172.20.0.1 - - [15/Sep/2019:18:13:36 +0000] "GET /admin/admin/vendor/bootstrap-daterangepicker-bs2.css HTTP/1.1" 200 0 "http://localhost:8080/admin/airflow/tree?dag_id=tutorial" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100 Safari/537.36"
webserver_1 | 172.20.0.1 - - [15/Sep/2019:18:13:36 +0000] "GET /admin/admin/vendor/bootstrap-daterangepicker/daterangepicker.js?v=1.3.22 HTTP/1.1" 200 0 "http://localhost:8080/admin/airflow/tree?dag_id=tutorial" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100 Safari/537.36"
webserver_1 | 172.20.0.1 - - [15/Sep/2019:18:13:36 +0000] "GET /admin/admin/js/form.js?v=1.0.1 HTTP/1.1" 200 0 "http://localhost:8080/admin/airflow/tree?dag_id=tutorial" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100 Safari/537.36"
webserver_1 | 172.20.0.1 - - [15/Sep/2019:18:13:36 +0000] "GET /static/tree.css HTTP/1.1" 200 0 "http://localhost:8080/admin/airflow/tree?dag_id=tutorial" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100 Safari/537.36"
webserver_1 | 172.20.0.1 - - [15/Sep/2019:18:13:38 +0000] "GET /admin/airflow/graph?dag_id=tutorial&execution_date= HTTP/1.1" 200 9219 "http://localhost:8080/admin/airflow/tree?dag_id=tutorial" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100 Safari/537.36"
webserver_1 | 172.20.0.1 - - [15/Sep/2019:18:13:38 +0000] "GET /static/dagre.css HTTP/1.1" 200 0 "http://localhost:8080/admin/airflow/graph?dag_id=tutorial&execution_date=" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100 Safari/537.36"
webserver_1 | 172.20.0.1 - - [15/Sep/2019:18:13:38 +0000] "GET /static/graph.css HTTP/1.1" 200 0 "http://localhost:8080/admin/airflow/graph?dag_id=tutorial&execution_date=" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100 Safari/537.36"
webserver_1 | 172.20.0.1 - - [15/Sep/2019:18:13:38 +0000] "GET /static/dagre-d3.js HTTP/1.1" 200 0 "http://localhost:8080/admin/airflow/graph?dag_id=tutorial&execution_date=" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100 Safari/537.36"
webserver_1 | 172.20.0.1 - - [15/Sep/2019:18:13:40 +0000] "GET /admin/ HTTP/1.1" 200 27710 "http://localhost:8080/admin/airflow/graph?dag_id=tutorial&execution_date=" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100 Safari/537.36"
webserver_1 | 172.20.0.1 - - [15/Sep/2019:18:13:40 +0000] "GET /admin/airflow/dag_stats HTTP/1.1" 200 442 "http://localhost:8080/admin/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100 Safari/537.36"
webserver_1 | 172.20.0.1 - - [15/Sep/2019:18:13:40 +0000] "GET /admin/airflow/blocked HTTP/1.1" 200 2 "http://localhost:8080/admin/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100 Safari/537.36"
webserver_1 | 172.20.0.1 - - [15/Sep/2019:18:13:41 +0000] "GET /admin/airflow/task_stats HTTP/1.1" 200 1445 "http://localhost:8080/admin/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100 Safari/537.36"
webserver_1 | [2019-09-15 18:13:52 +0000] [162] [INFO] Handling signal: ttin
webserver_1 | [2019-09-15 18:13:52 +0000] [1000] [INFO] Booting worker with pid: 1000
webserver_1 | [2019-09-15 18:13:52,648] {{__init__.py:51}} INFO - Using executor LocalExecutor
webserver_1 | [2019-09-15 18:13:52,962] {{dagbag.py:90}} INFO - Filling up the DagBag from /usr/local/airflow/dags
webserver_1 | [2019-09-15 18:13:53 +0000] [162] [INFO] Handling signal: ttou
webserver_1 | [2019-09-15 18:13:53 +0000] [343] [INFO] Worker exiting (pid: 343)
```

Agora foi

Acesse pelo navegador:

<http://localhost:8080/>

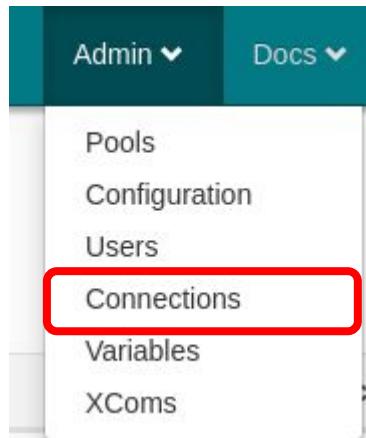


A close-up photograph of a vintage-style teal robot toy. The robot has a rounded rectangular head with two large, circular, light-colored eyes with dark pupils. It has a simple triangular mouth area with a red base plate featuring white horizontal stripes. A black coiled antenna is attached to the top of its head. A red push-button is located on the left side of the head. The body is teal with a pattern of small white circles along the bottom edge. The robot is positioned on the left side of the frame.

7. SELECT test FROM oficina

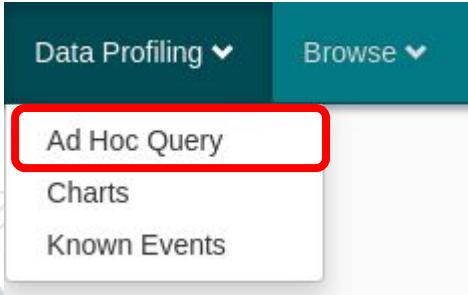
SELECTzinho básico pra conferir se está tudo ok

1. Criar conexão



Conn Id *	db_oficina
Conn Type	Postgres
Host	postgres
Schema	exercicios_airflow
Login	airflow
Password airflow
Port	5432

2. Executar a Query



db_oficina	Run!	.csv
1 select * from indigenas		

Origem dos dados

◎ Terras Indígenas - estados - 2017.csv

◎ [http://dados.gov.br/dataset/sistema-nacional-de-i
nformacoes-florestais-snif/resource/6086a36e-d5d
c-4fea-b2e0-a9d1d3229867](http://dados.gov.br/dataset/sistema-nacional-de-informacoes-florestais-snif/resource/6086a36e-d5dc-4fea-b2e0-a9d1d3229867)





8.
[break] Ufa,
DESCANSO
(20 min.)

A photograph of a lioness with a light brown coat resting its head on a large, textured tree trunk. She is standing on a wooden deck. A dashed white circle highlights her front right leg and paw.

8.
[break] Ufa,
DESCANSO
(20 min.)

9.

Hello World: Seu primeiro robô (ou não)



Orientações

1. Edite o arquivo
desafio/desafio_hello_world_dag.py seguindo as instruções nos comentários
2. A solução com o script python está em
solucao/solucao_hello_world_dag.py.sol, mas
guarde esse segredo ;P

Estrutura de uma DAG

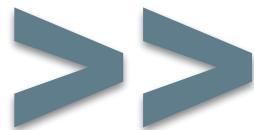
```
# Libraries  
# Tudo ok  
...  
  
# DAG  
# Definir parâmetros  
...  
  
# Tasks  
# Set nome e comando bash  
...  
  
# Orchestration  
# Ordem de execução  
...
```

Prática:

Implementar uma DAG com 2 operadores Bash que imprimem "Hello" e "World!"

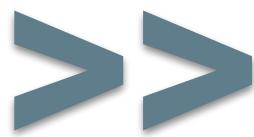
Para dar o Play

1. No painel do Airflow, ligue a DAG:



	<i>i</i>	DAG	Schedule
		tutorial	1 day, 0:00:00

2. No painel, nos botões “Links”, acione o “Trigger Dag”:



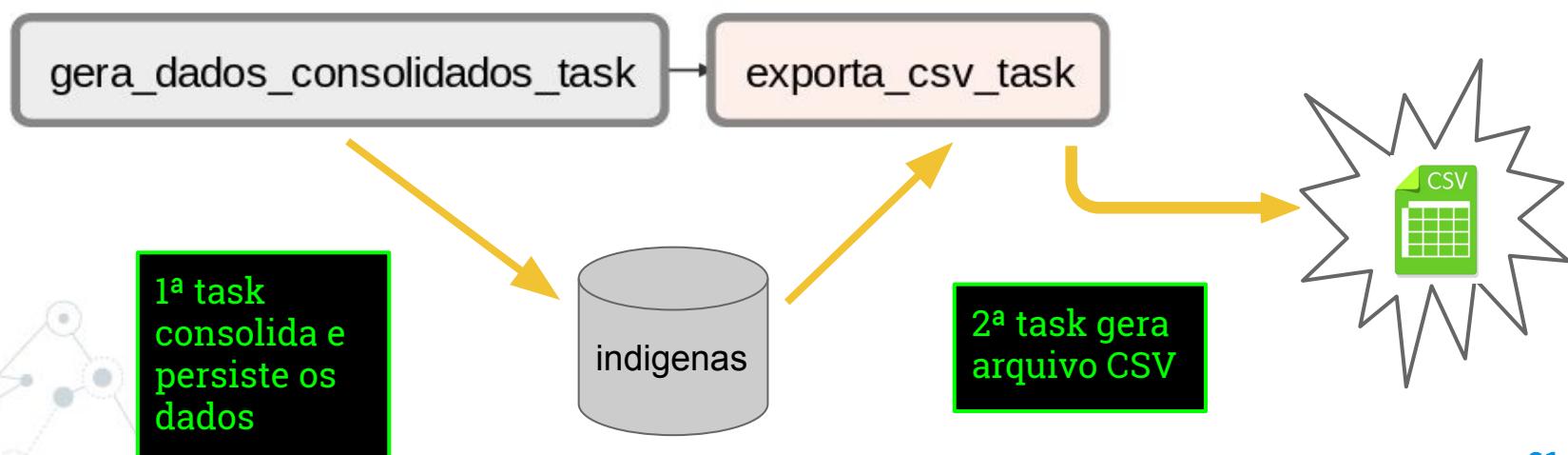
10.

Baby Steps:
Implementação
de uma DAG pra
chamar de sua



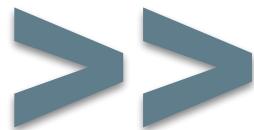
Desafio firmeza

1. Edite o arquivo **desafio/desafio_baby_steps_dag.py** seguindo as instruções nos comentários
2. Edite o arquivo **sql_query/desafio_query_uf_superficie_total.sql** seguindo as instruções nos comentários
3. A solução com o script python está em **solucao/solucao_baby_steps_dag.py.sol**, mas guarde esse segredo ;P



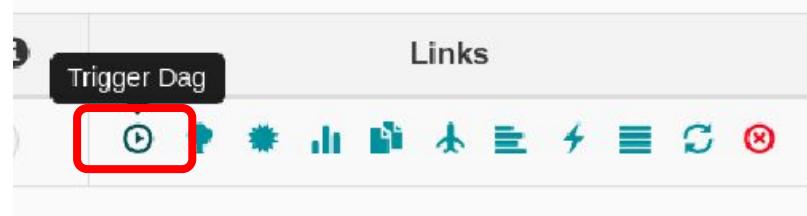
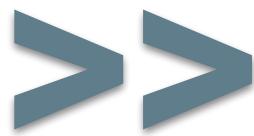
Para dar o Play

1. No painel do Airflow, ligue a DAG:



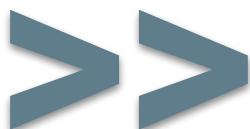
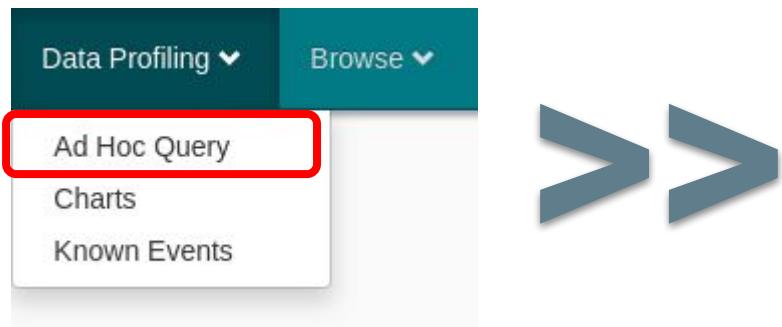
	<i>i</i>	DAG	Schedule
		tutorial	1 day, 0:00:00

2. No painel, nos botões “Links”, acione o “Trigger Dag”:



Se pá deu certo

1. Verificar se os dados foram gerados. Execute a query:



A screenshot of a database query editor. The top bar shows 'db_oficina', 'Run!', and '.csv'. The main area contains a query: '1 select * from uf_superficie_total'.

2. Verificar se o arquivo CSV foi gerado. Acesse a pasta “**export/**” e procure pelo arquivo **.csv**

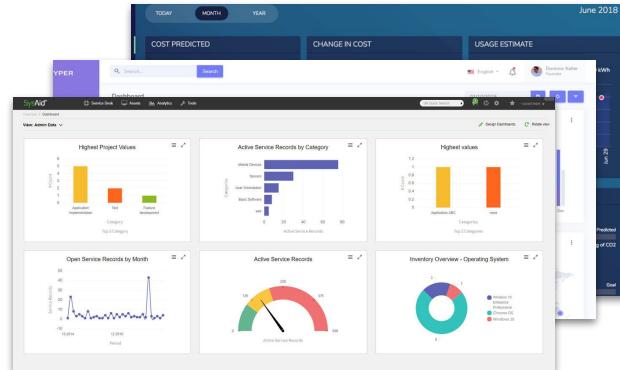


11.

Cataventos do ME

Somos a Coordenação-Geral de Gestão da Informação/SEGES

- Fazemos apoio à gestão estratégica
- Painéis informacionais
- Integrando dados estruturantes
- Mais de 2 anos produzindo



Acumulamos uma dívida técnica grande no backend.



Convertendo o Chefe

PROJETOS ESTRATÉGICOS

Diagnóstico e Monitoramento do Projeto de Transformação Institucional

Evolução e Fortalecimento do Sistema Informatizado SIORG



PORFÓLIO DE PROJETOS

Evolução do SIORG

Capacitação EAD
SIORG

Painel de Gestão
Governamental

Painéis de Carreiras e
Central de Atendimento

Data Lake

Evolução do Portal e da
API do SIORG

Central de Atendimento
SIORG

Raio-X

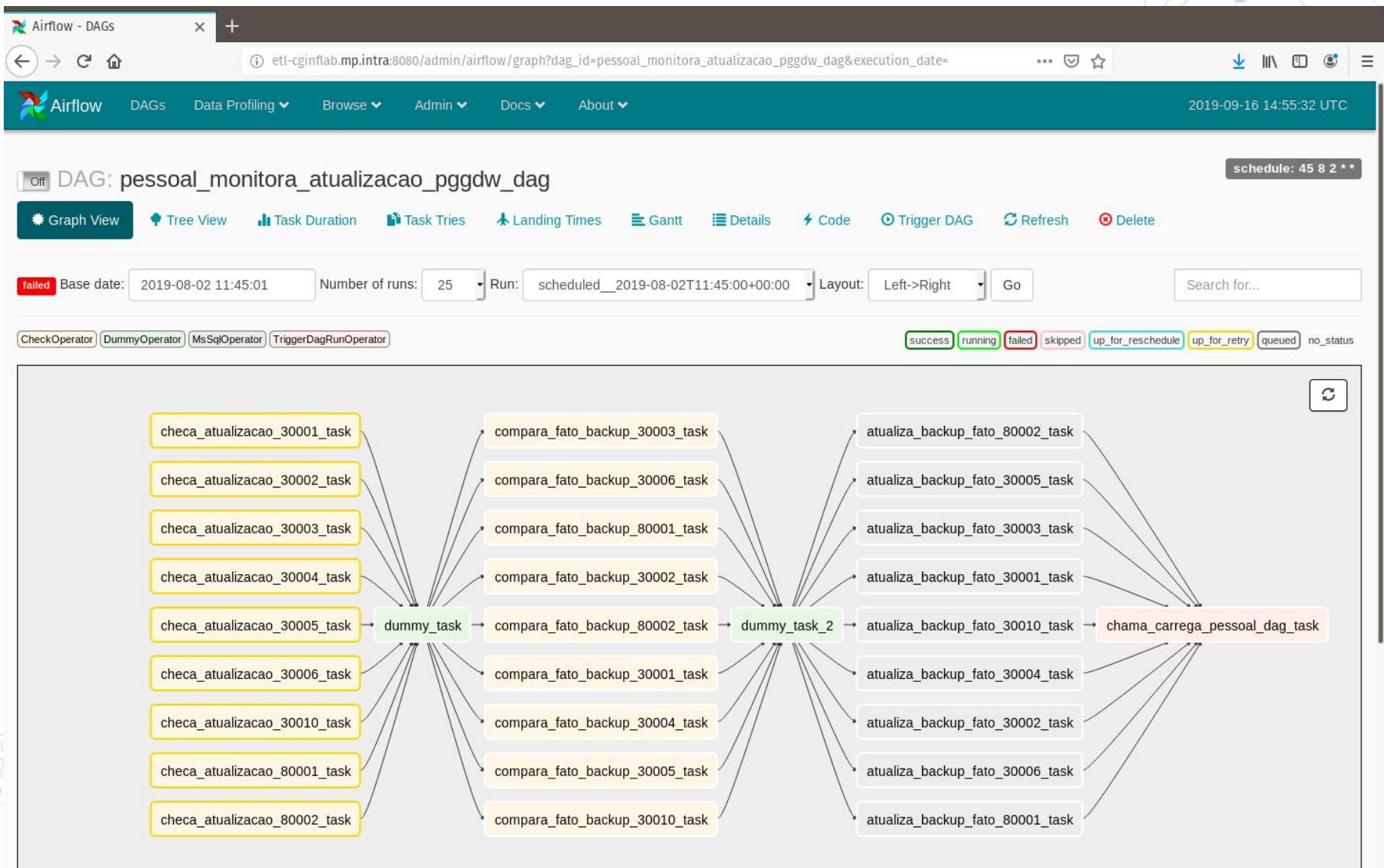
Integração
SIORG – SIGEPE

Política de Segurança
do SIORG

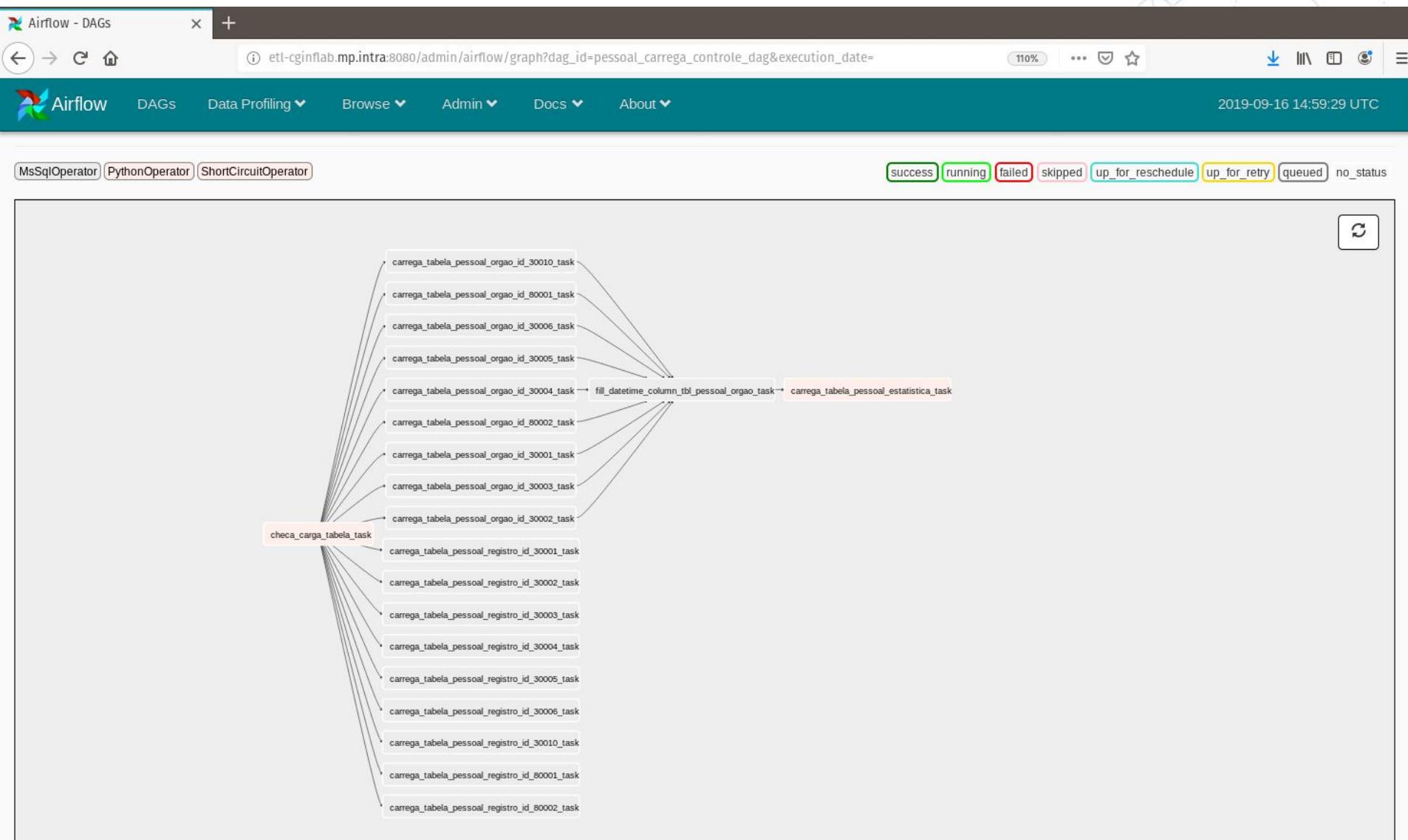
Boletim Periódico

Automação e
Monitoramento do Fluxo
de Dados

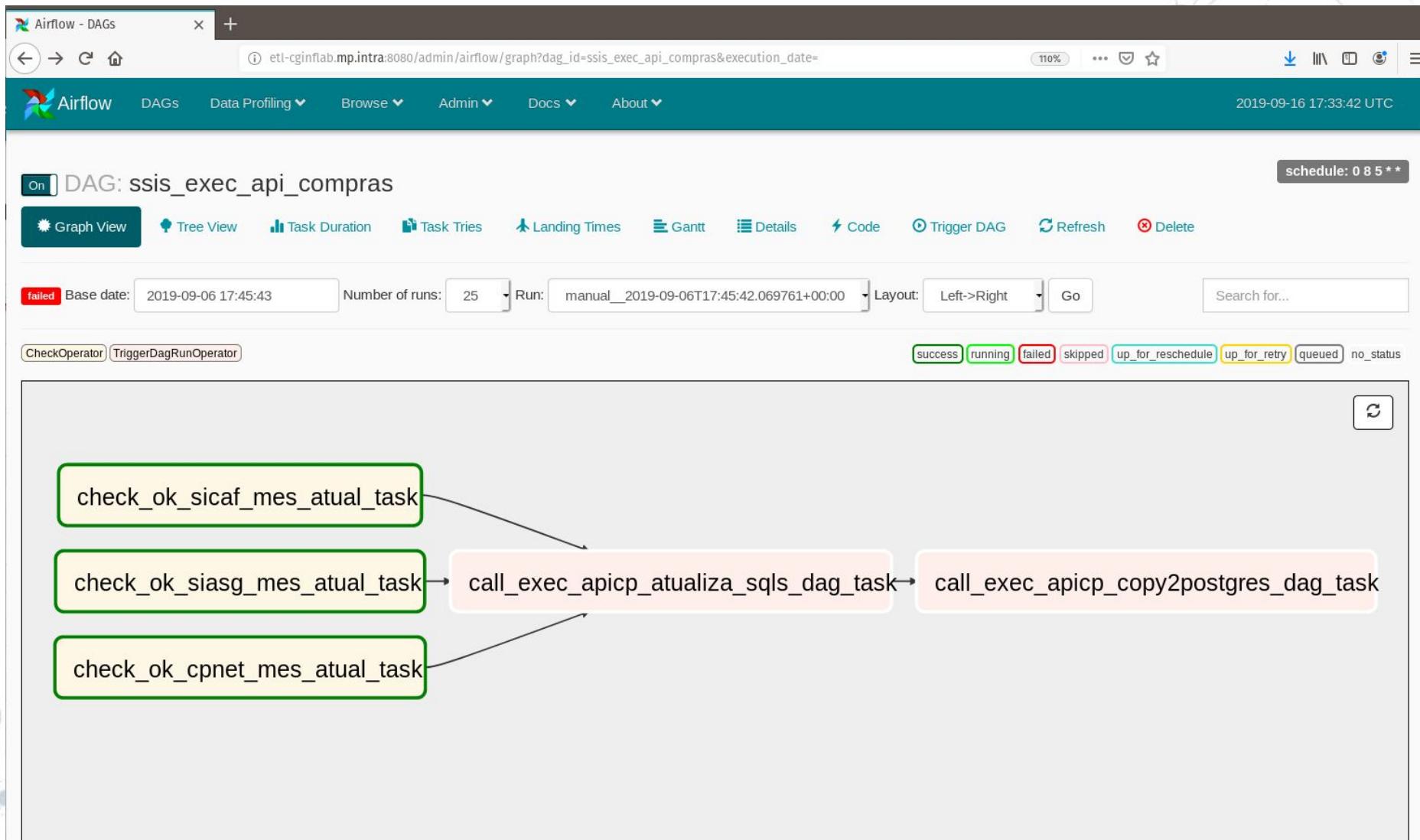
Monitoramento e visualização do ETL Legado



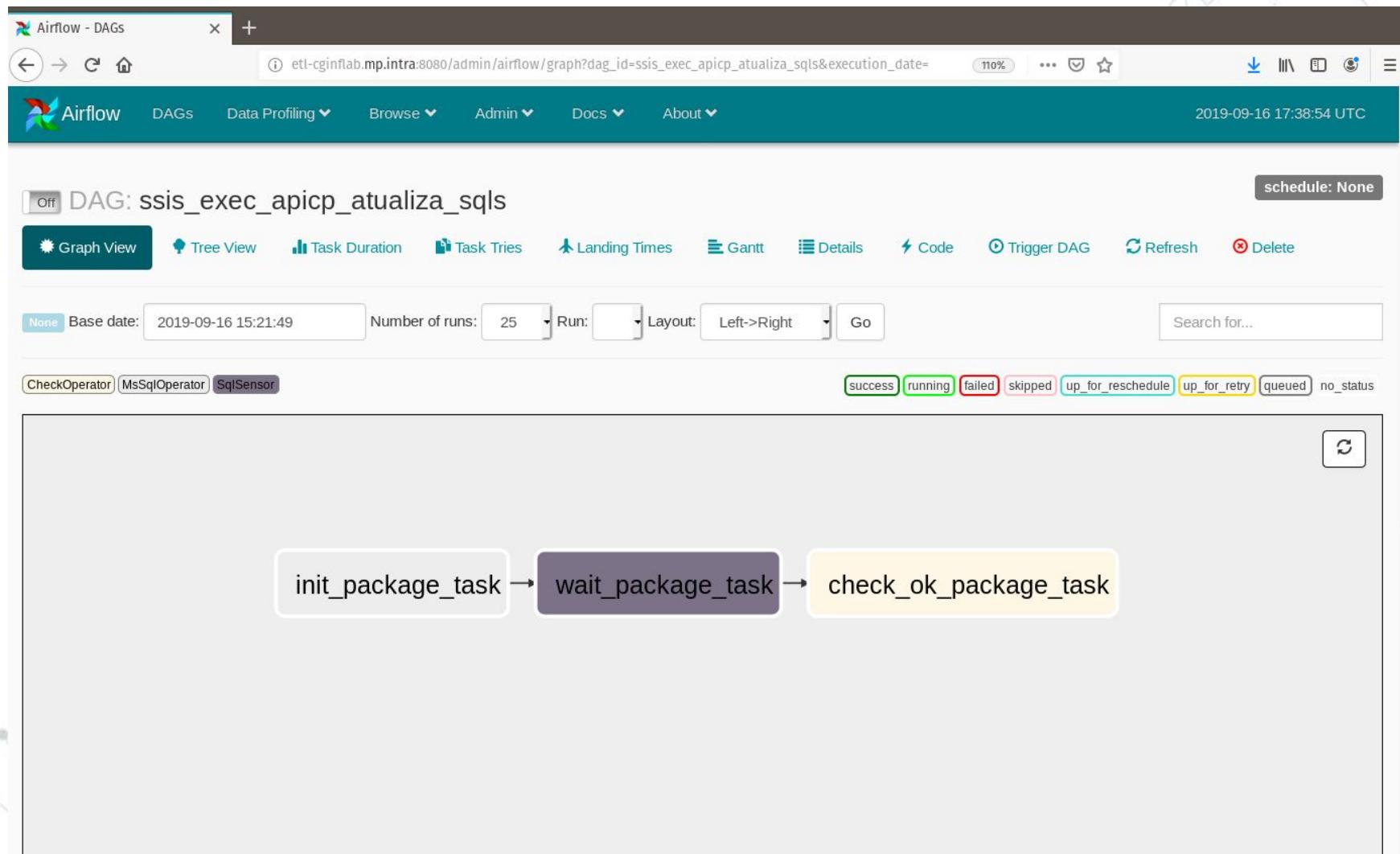
Monitoramento e visualização do ETL Legado



Orquestração da carga da API de Compras



DAG secundária





12.

Dicas e Truques (quem souber me conta)

Reflexões de arquitetura

- Papel do Airflow como Orquestrador e não como motor de ETL
- Uma DAG enorme **vs** Várias pequenas DAGs
- Caminhos seguros para transição
- Geração automática de DAGs
- PythonOperator **vs** Custom Operators

Treinamentos

- ◎ [The Complete Hands-On Course to Master Apache Airflow - Udemy/Marc Lamberti](#)
- ◎ [Apache Airflow Tutorials - Apply Datascience](#)

Nuvens com Airflow PAAS

- ◎ Google Cloud Composer
<https://cloud.google.com/composer/>
- ◎ Astronomer.io

Manual e manuais

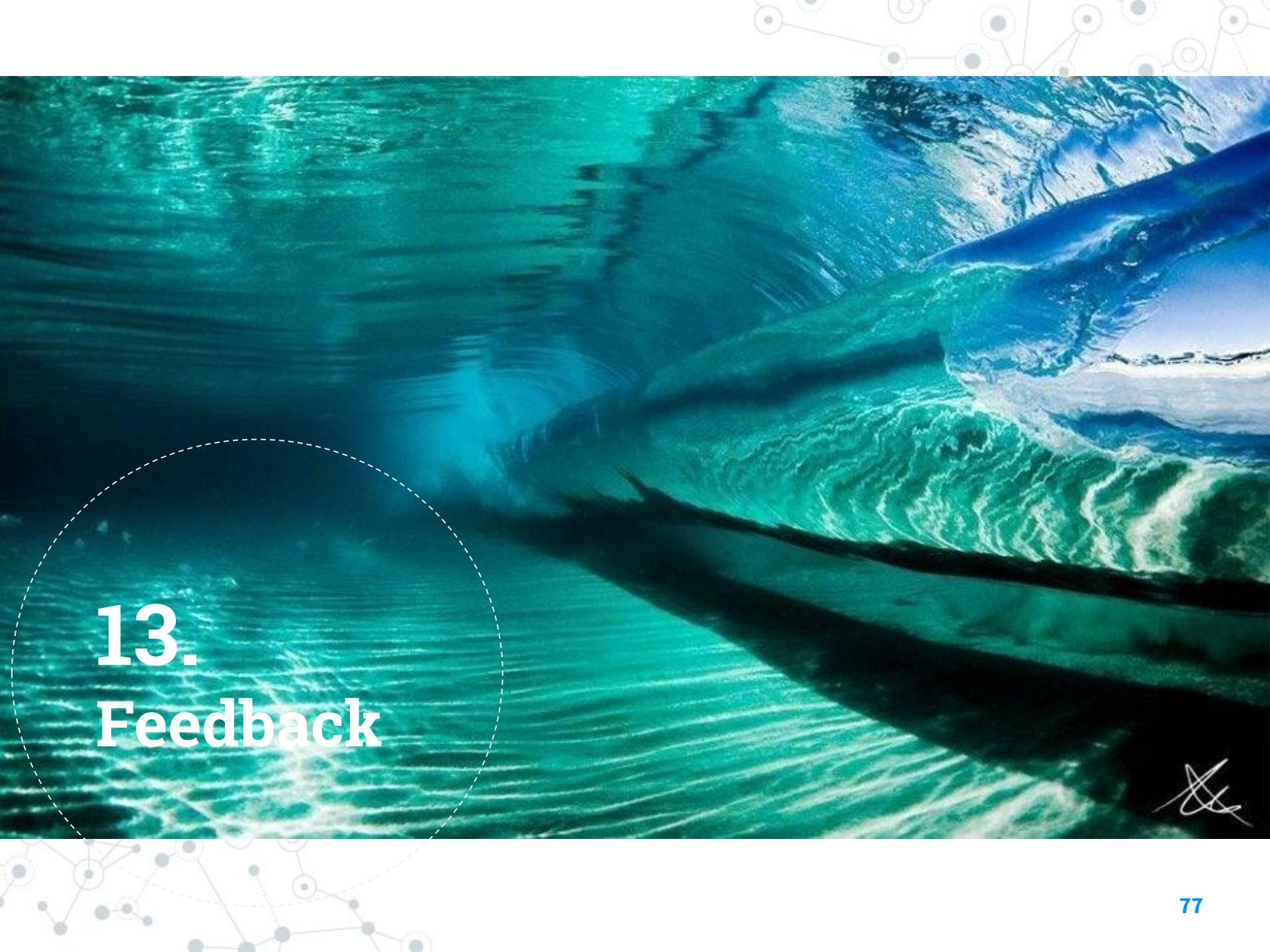
- [Manual Oficial Apache Airflow](#)
- [Github Apache Airflow](#)
- [Puckel/docker-airflow \(Github\)](#)
- [ETL best practices with Airflow documentation site - Gerard Toonstra](#)
- [Best practices with Airflow- an open source platform for workflows & schedules - Maxime Beauchemin](#)
- [Hands-On Big Data Online Courses - Marc Lamberti](#)

ecodados.economia.gov.br

Para facilitar o lançamento do seu novo site, você está no modo de bootstrap. Todos os novos usuários receberão o nível de confiança 1 e terão os e-mails diários de resumo ativados. Isso será desativado automaticamente quando 100 usuários se registrarem.

todas as categorias ▶ Recente Melhores Categorias + Novo Tópico

Tópico	Respostas	Visualizações	Atividade
Scheduling Notebooks at Netflix medium.com ■ Artigos e Publicações jupyter, airflow, notebooks	2	7	4d
Voltamos! ou: renovação do certificado digital ■ Equipe downtime, lets-encrypt	3	6	6d
Base de Acórdãos TCU ■ Projetos e Experimentos	2	11	6d
Voltamos ao ar! ■ Meta downtime, lets-encrypt	0	14	7d
Precisamos realmente de tantos Cientistas de Dados? ■ Artigos e Publicações airflow	0	32	18d
Surfando na crista da onda de visualização de dados (Superset e Metabase) ■ Artigos e Publicações visualização, dashboard	7	65	19d
On the Philosophical Foundations of Conceptual Models ■ Artigos e Publicações modelagem-conceptual	1	19	21d



13. Feedback

Feedback



<https://bit.ly/2meiZvs>

*Com o cadastro do e-mail mandamos
esses slides 😊*

Inté!

Nitai Bezerra

nitai.silva

@planejamento.gov.br

t. 2020-1551



@nitaís

Vitor Bellini

vitor.bellini

@planejamento.gov.br

t. 2020-1504



@podefalar

