



Faça login em Medium com o Google



Gustavo Santos

progustavosantos@gmail.com

Continuar como Gustavo

Certificação Airflow



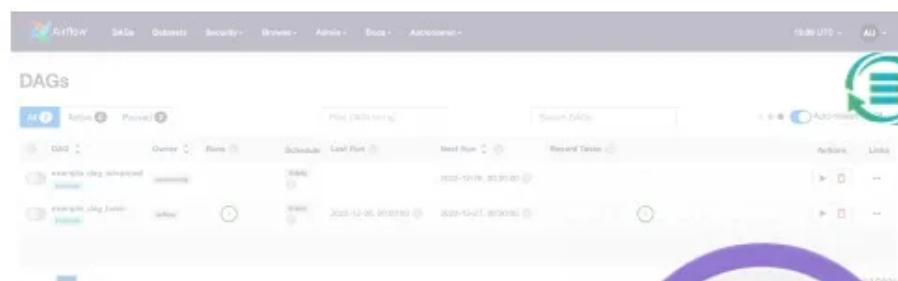
Anselmo Borges · Follow

Published in Rescue Point

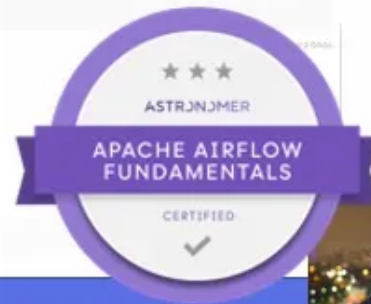
6 min read · Jan 4



Share



Apache
Airflow



Anselmo Borges
DataOps Engineer
Microsoft MCT

Certificação Airflow Fundamentals (3/3)

Fechando o cronograma de materiais necessários para a certificação Airflow fundamentals, agora que já sou certificado!

Ultima parte do material de certificação em Apache Airflow

Fiz a prova semana passada, antes do ano novo, passei e não vou mentir que deu uma preguiça vir aqui e escrever esse conteúdo mas compromisso é compromisso.

Nessa última parte vamos ver:

- Compartilhando dados entre tarefas no Airflow
- Trabalhando com falhas, como agir
- Parâmetros sobre concorrência e executors

- Como funciona a prova e onde fazer
- Agradecimentos

Compartilhando dados entre tarefas no Airflow

Vamos supor que dentro do nosso código a gente precise compartilhar informações entre as tasks do Airflow, o caminho para fazer isso é usando as chamadas Xcoms (Cross Communication).

Com base nos nossos cenários anteriores vamos simular o compartilhamento de dados específicos entre as tasks "baixando_dados" e "esperando_dados". A task "baixando_dados" chama uma function python de nome "_baixando_dados", vamos altera-la a fim de simular esse compartilhamento de informações.

Coloquei no final da função um "return 42" conforme podem ver no código abaixo:

```
def _baixando_dados(**kwargs):  
    with open('/tmp/meu_arquivo.txt', 'w') as f:  
        f.write('meus dados')  
    return 42
```

Esse valor é automaticamente encapsulado em uma Xcom, salvei o arquivo e iniciei a DAG "simple_dag", voltando para a UI do Airflow coloco pra rodar e verifico na aba Admin >> Xcoms conforme gif abaixo que a informação 42 é gerada pela tarefa "baixando_dados".

The screenshot displays the Airflow DAGs interface. At the top, there's a navigation bar with 'Airflow' and various menu items like 'DAGs', 'Datasets', 'Security', 'Browse', 'Admin', 'Docs', and 'Astronomer'. Below this, the 'DAGs' section is active, showing a list of DAGs. The table has columns: DAG, Owner, Runs, Schedule, Last Run, Next Run, and Recent Tasks. The 'simple_dag' is selected, showing 4 runs and 3 recent tasks. The 'dag_do_anselmo' is also visible with 2 runs and 3 recent tasks. The 'example_dag_advanced' and 'example_dag_basic' are also listed. The bottom of the screen shows 'Astronomer Runtime 7.1.0 based on Airflow 2.5.0+astro.1'.

Teste de Xcoms

Se você reparar o valor 42 está vinculado a uma key de nome "return_value", ou seja, um registro chave/valor onde todas as outras informações eu conseguiria pegar também, como nome da DAG, nome da DAG, timestamp e etc.

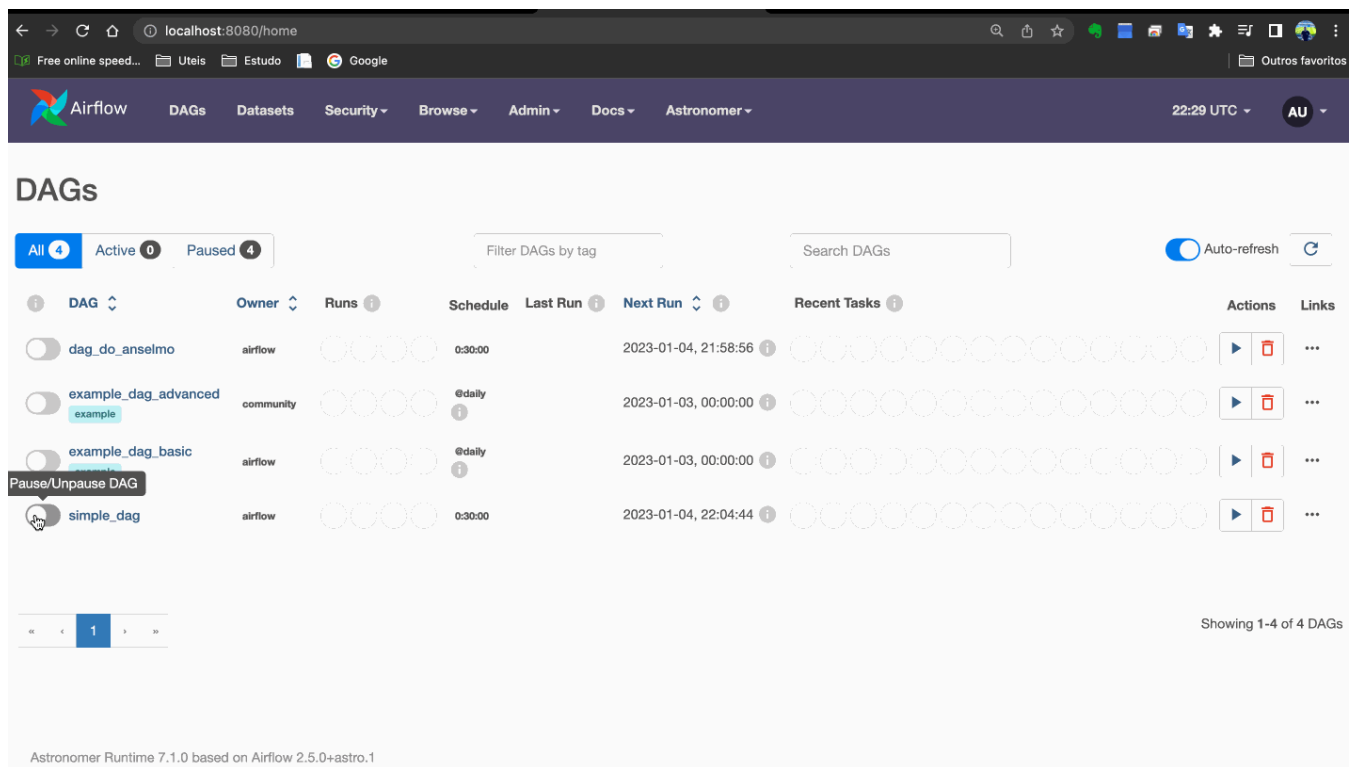
Vamos agora na segunda task de nome "esperando_dados" usar o parâmetro xcom_pull mas antes vamos criar uma nova function de nome "_usando_xcom" e aí sim atribuir a task "esperando_dados" usando o python_callable.

```
def _usando_xcom(ti):
    my_xcom = ti.xcom_pull(key='return_value', task_ids=['baixando_dados'])
    print(my_xcom)
```

Agora atribuímos ao Task "esperando_dados"

```
esperando_dados = PythonOperator(
    task_id='esperando_dados',
    python_callable=_usando_xcom
)
```

Eu alterei a task 2 para PythonOperator apenas para fazer o teste, não é o ideal a se fazer. Salvei e coloquei pra rodar, vamos ver como se sai.



The screenshot shows the Apache Airflow web interface at localhost:8080/home. The top navigation bar includes links for DAGs, Datasets, Security, Browse, Admin, Docs, and Astronomer. The main content area is titled 'DAGs' and displays a table of DAGs. The table has columns for DAG, Owner, Runs, Schedule, Last Run, Next Run, Recent Tasks, Actions, and Links. Four DAGs are listed: dag_do_anselmo, example_dag_advanced, example_dag_basic, and simple_dag. The simple_dag is highlighted with a mouse cursor over the 'Pause/Unpause DAG' button. The interface also includes a search bar, a filter by tag dropdown, and an auto-refresh toggle.

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions	Links
dag_do_anselmo	airflow	0	0:30:00		2023-01-04, 21:58:56		[Play] [Stop] [More]	
example_dag_advanced	community	0	@daily		2023-01-03, 00:00:00		[Play] [Stop] [More]	
example_dag_basic	airflow	0	@daily		2023-01-03, 00:00:00		[Play] [Stop] [More]	
simple_dag	airflow	0	0:30:00		2023-01-04, 22:04:44		[Play] [Stop] [More]	

Coloquei pra rodar e validei o log da task 2 e vemos o 42 lá

Usamos o 42 mas poderia ser qualquer atributo e eu poderia usar também como no exemplo abaixo, em vez do return, o parâmetro "xcom_push" e seto um valor pra pegar no meu caso será 43, vou alterar a função "_baixando_dados" com essas informações.

```
def _baixando_dados(ti, **kwargs):  
    with open('/tmp/meu_arquivo.txt', 'w') as f:  
        f.write('meus dados')  
        ti.xcom_push(key='key_anselmo', value=43)  
  
def _usando_xcom(ti):  
    my_xcom = ti.xcom_pull(key='key_anselmo', task_ids=['baixando_dados'])  
    print(my_xcom)
```

Note que criei uma chave chamada key_anselmo no push e no pull estou colhendo ela, mas naquele esquema anterior de cada função em uma task, vai dar pra ver no gif.

Usando o xcom_push e xcom_pull com a key_anselmo

Trabalhando com falhas no Airflow

Vamos simular uma falha e ver que parâmetros podemos usar para usar um tipo de notificação se necessário. Na terceira task vamos alterar o exit 0 que é uma saída com sucesso para exit 1 que simula um erro.

```
...
default_args = {
    'retry': 2,
    'retry_interval': timedelta(seconds=10)
}

...
processando_dados = BashOperator(
    task_id='processando_dados',
    bash_command='exit 1'
)
```

Pra ficar mais rápido o erro, mudei o retry pra 2 e o intervalo de retry pra 10 segundos. Salve e vamos rodar.

The screenshot displays the Airflow web interface at localhost:8080/home. The top navigation bar includes links for DAGs, Datasets, Security, Browse, Admin, Docs, and Astronomer. The main section is titled 'DAGs' and features a filter for 'All' (4) DAGs, with 'Active' (0) and 'Paused' (4) counts. A search bar and an 'Auto-refresh' toggle are also present. The table lists four DAGs: 'dag_do_anselmo', 'example_dag_advanced', 'example_dag_basic', and 'simple_dag'. The 'simple_dag' is selected, showing its 'Runs' column with a status of 'Failed' (red circle with an 'x'). The 'Recent Tasks' column for 'simple_dag' shows three tasks, with the third task (task 3) highlighted in blue, indicating it is the current task being viewed. The bottom of the page shows the version information: 'Astronomer Runtime 7.1.0 based on Airflow 2.5.0+astro.1'.

Simulamos um erro na 3a task da DAG

Bom mas e ae? Toda vez vou ter que olhar? Quem vai avisar? Então, existem alguns parâmetros que você pode configurar na task pra que ele envie uma notificação, vamos começar pela mais simples, email.

- **email_on_failure:** Quando seto esse parâmetro pra TRUE, caso sua DAG falhe ele manda um email caso você tenha os outros parametros configurados e o servidor SMTP configurado no cluster Airflow (outra conversa, outro dia).
- **email_on_retry:** Mesmo esquema do de cima mas envia só em caso de retry.
- **email:** Você coloca o email que vai enviar, por exemplo, admin@rescuepoint.com.br

Vamos supor que você quer algo mais incrementado (acredito que você queira, pois ninguém lê emails nem quando backup falha, desde quando eu era DBA Oracle e finge que não leu isso).

Hoje existe outros métodos de notificação como canal no Teams, no Slack e alguns até notificando no Whatsapp numa gravidade maior, via webhook, uma requisição API simples pra onde você deseja notificar, até como um sistema de chamados. Você pode criar uma função python que faça o que você precisa e chama-la através do parâmetro **on_failure_callback**, vou usar um exemplo com uma funçãozinha tosca mas pense que pode ser o que vc quiser.

```

## Função
def _funcao_tosca(context):
    print("deu erro aqui!")
    print(context)

...
## Task chamando a função no caso de falha:
processando_dados = BashOperator(
    task_id='processando_dados',
    bash_command='exit 1',
    on_failure_callback=_funcao_tosca
)

```

No script acima criei uma função que vai fazer prints simples no caso de falha chamado pela task 3 que estamos testando simulando erros.

The screenshot shows the Apache Airflow web interface at localhost:8080/home. The top navigation bar includes links for DAGs, Datasets, Security, Browse, Admin, Docs, and Astronomer. The main section is titled 'DAGs' and displays a table of DAGs. The table has columns for DAG, Owner, Runs, Schedule, Last Run, Next Run, Recent Tasks, and Action. The 'simple_dag' is selected, and its task ID '3' is circled in red in the 'Recent Tasks' column. The interface also shows filters for DAGs by tag and a search bar. The bottom status bar indicates 'Astronomer Runtime 7.1.0 based on Airflow 2.5.0+astro.1'.

Funcionou a tosquite

Mas deu pra entender que pode fazer o que eu jogar na função, inclusive um plano de correção temporário, sei lá (vale mais a pena arrumar a DAG).

Parâmetros de concorrência

Recaptulando rapidão, existem alguns tipos de executors:

- **Sequenciais:** Esse que estamos usando aqui, pois temos um ambiente bem limitado e de um único node, ou seja, todas as tasks rodam de forma sequencial

no mesmo node de executor (é o padrão)

- **Local:** Vamos supor que tenho um node só mas ele é parrudão, com bastante memória e bastante processamento, quero rodar paralelo por minha conta e risco. Esse é o modelo de executor que você seta na configuração do seu Airflow.
- **Celery:** Tenho um cluster fisico, sem ser container nem nada, composto por 5 maquinas por exemplo, separo webserver e metadata em uma, scheduler e um serviço de fila como **RabbitMQ** em outra (já explico mas não vou entrar em detalhes) e as outras 3 passam a ser nodes, que vão no serviço de fila e pega o que tem pra executar, esse nodes são chamados **worker nodes**.
- **Kubernetes:** Mesmo esquema de cluster citado acima mas usando PODs Kubernetes ao invés de nodes fisicos.

Usando os modelos em cluster, conforme sua necessidade de paralelizar mais tasks aumentar, basta adicionar novos worker nodes e vai conseguir dar conta da demanda, mas tem alguns parâmetros que podem ajudar também.

- **Parallelism:** O numero de tasks que você pode executat em paralelo no seu cluster Airflow inteiro. O valor padrão é 32, ou seja, por padrão é permitido você executar 32 tasks em paralelo no seu cluster.
- **DAG_concurrency:** Numero de tasks que podem ser executadas em paralelo de todas as suas DAGRuns. O Valor padrão é 16.
- **max_active_runs_per_DAG:** Numero de DAGRuns que você pode ter rodando ao mesmo tempo. O valor padrão também é 16.
- **max_active_runs:** Numero de DAGRuns setado em uma DAG específica (vimos isso na parte 1)
- **concurrency:** Numero de tasks em paralelo que consigo rodar numa DAG especificada. Funciona parecido com o de cima, não é um parâmetro global como os 3 primeiros.

Como fazer a prova

Se você chegou até aqui e entendeu tudo, PARABÉNS!

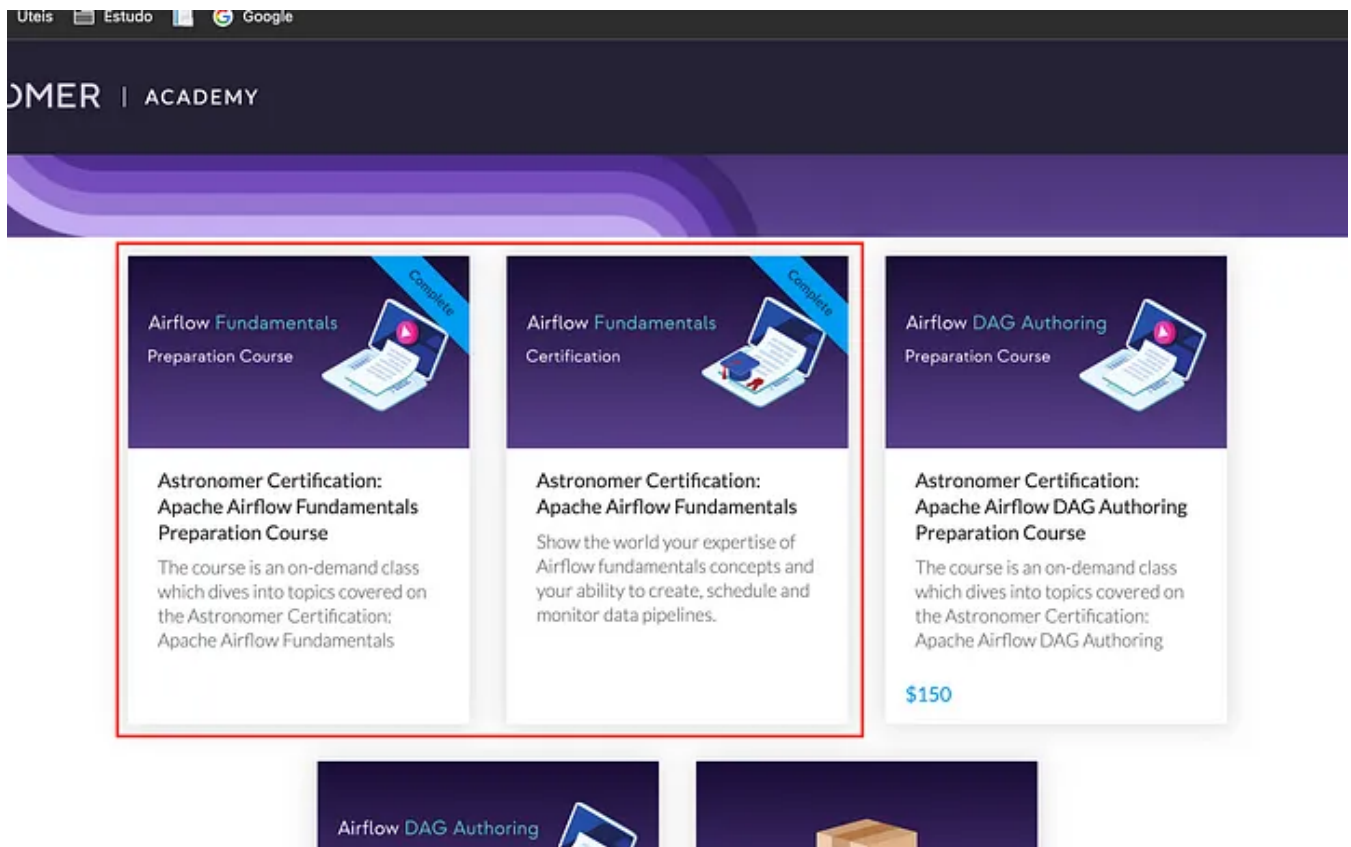




Alegria

Agora vamos ver como fazer a prova.

1. Crie uma conta no site da Astronomer: <https://academy.astronomer.io/>
2. Você vai selecionar esses cursos aqui nessa pagina:
<https://academy.astronomer.io/page/astronomer-certification>



O segundo é a prova, de graça

A prova tem 75 questões e 120 minutos pra fazer, exige um acerto de pelo menos 70% dela e as perguntas são bem baseadas nesses 3 posts, se você fez tudo aqui não tem como dar ruim.

Se passar me marca lá no LinkedIn pra dar essa moral e marca também o autor do curso Mark Lamberti, que alias tem um canal no Youtube falando muito mais sobre Airflow caso você vá usar mais a ferramenta e a didática do cara é incrível e por menos que eu fale inglês é bem "entendível", rs.

Data with Marc

As a 30-year-old data engineer and trainer, best selling author on Udemy about Apache Airflow, and Head of Customer...

www.youtube.com

Bom é isso, espero que te ajude, se pintar alguma dúvida e eu puder ajudar me chama lá no LinkedIn e nos falamos.

Deixa a palminha ae, se inscreve aqui e no Youtube pra dar aquela força.

Valew e até o próximo!

Anselmo Borges.

Apache Airflow

Astronomer

Data Engineering

Certification

Rescue Point



Follow

Written by Anselmo Borges

491 Followers · Editor for Rescue Point

Bigdata Engineer, Cloud Architect, Nerd, Alcoholic, Brazilian JiuJitsu Black belt and hide and seek World champion.

More from Anselmo Borges and Rescue Point

Open in app ↗

Sign up

Sign In



Search Medium



Apache Airflow

Airflow Certification Fundamentals (2/3)

Nessa segunda parte vamos testar os acessos via CLI e REST API no Airflow, criar nossa primeira DAG já definindo agendamento, frequência, entenderemos um pouco mais sobre os DAGRuns, backfill, Operators e seus tipos e parâmetros.



Anselmo Borges
DataOps Engineer
Microsoft MCT



Anselmo Borges in Rescue Point

Certificação Airflow Fundamentals (Parte 2/3)

Continuando nosso material sobre a certificação básica em Apache Airflow da Astronomer, com um material bem rico com fotos, videos e...

9 min read · Dec 30, 2022



66



Certificação Airflow Fundamentals (Parte 1/2)
Um guia com um resumo do material para a certificação de Airflow da Astronomer, na primeira de 2 partes, como subir o ambiente, arquitetura do Airflow, overview sobre o Funcionamento e visualizações de suas DAGs de forma eficiente na UI.

Anselmo Borges
DataOps Engineer
Microsoft MCT



Anselmo Borges in Rescue Point

Certificação Airflow Fundamentals (Parte 1/2)

Me rendi ao Airflow, aproveitando meus estudos, criei uns resumos com base no curso da Astronomer, que pode te ajudar na certificação...

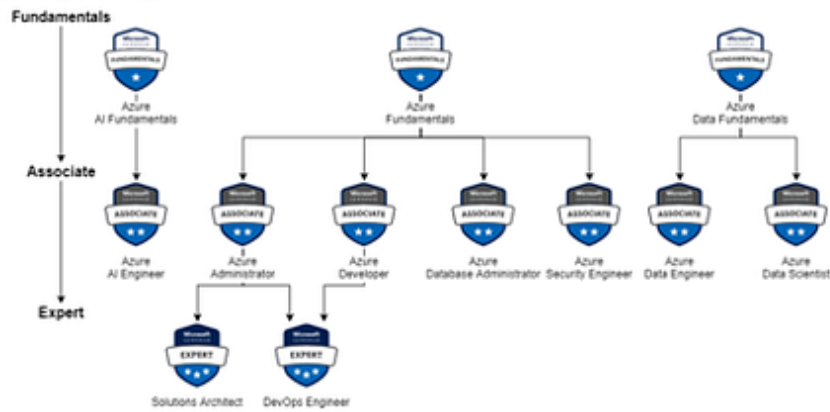
14 min read · Dec 27, 2022



148

1





Anselmo Borges
DataOps Engineer
Microsoft MCT

Certificações Microsoft Azure na faixa

Um overview sobre as certificações e como obter vouchers para as provas, que vão de 50 a 100% do valor da prova.

Anselmo Borges in Rescue Point

Como se certificar em Microsoft Azure Gratuitamente?

5 formas de arrumar vouchers de certificações em Microsoft Azure que podem variar de 50% a 100% dependendo da prova.

6 min read · May 31, 2022

75 2



Compilado certificação Airflow Fundamentals

Um compilado de 3 posts feitos com base no material da Astronomer que vão te ajudar a passar nessa prova que está de graça!



Anselmo Borges
Data Engineer Specialist
Microsoft MCT

Anselmo Borges in Rescue Point

Compilado Certificação Airflow

Compiladão dos 3 posts que te ajudam a tirar a certificação de Airflow Fundamentals da Astronomer, aproveite, a prova está de graça denovo!

2 min read · Aug 25



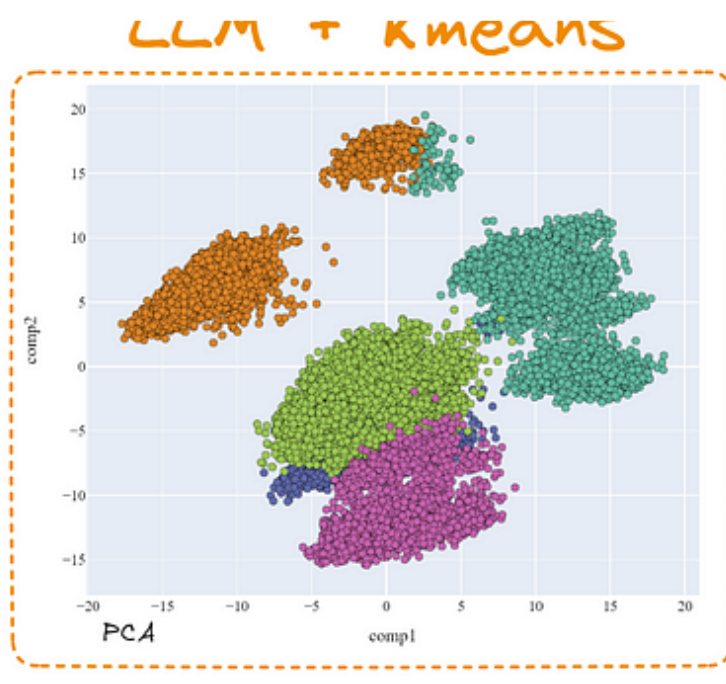
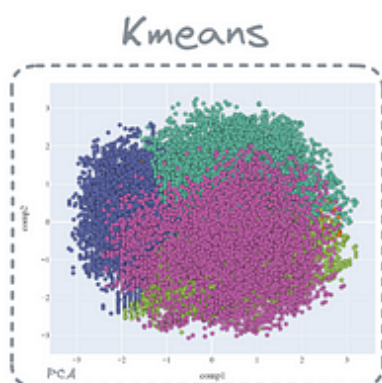
49



See all from Anselmo Borges

See all from Rescue Point

Recommended from Medium

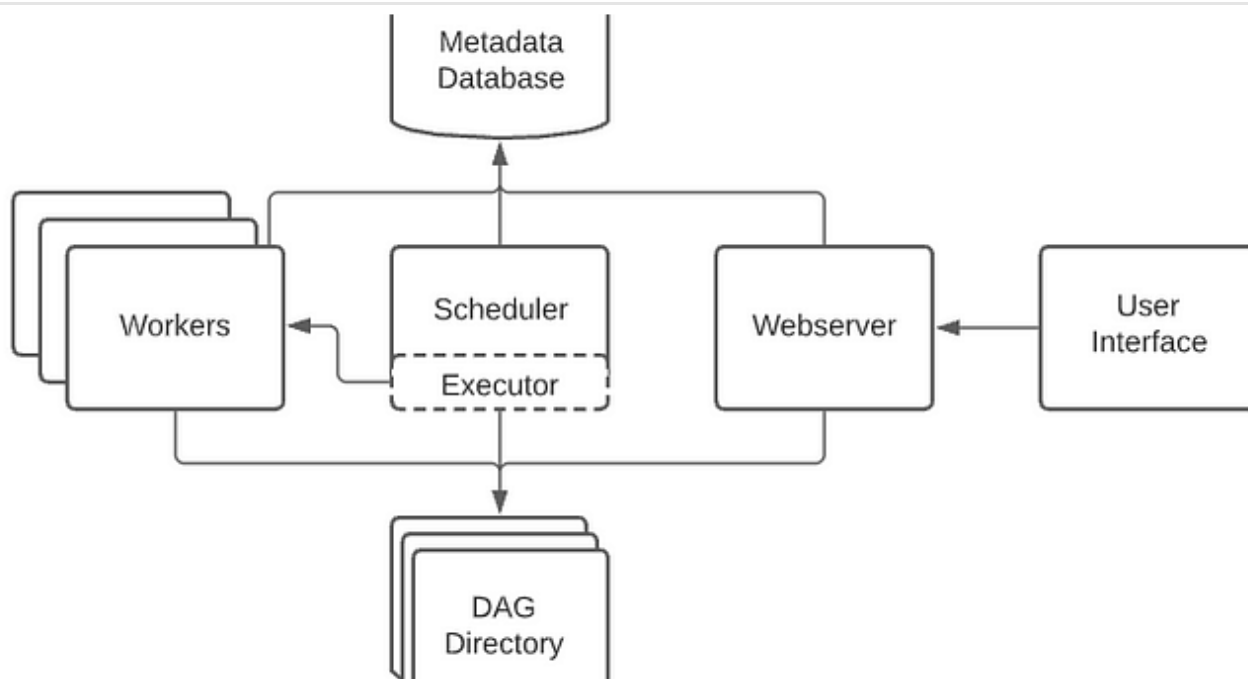


Damian Gil in Towards Data Science

Mastering Customer Segmentation with LLM

Unlock advanced customer segmentation techniques using LLMs, and improve your clustering models with advanced techniques

23 min read · 5 days ago



H Himanshu Kumar in Naukri Engineering

Scaling and Achieving High Availability (HA) in Airflow using Local Executor

Airflow is a workflow management system that lets the user define workflows programmatically and monitor them using a web interface...

4 min read · Jun 14

Lists



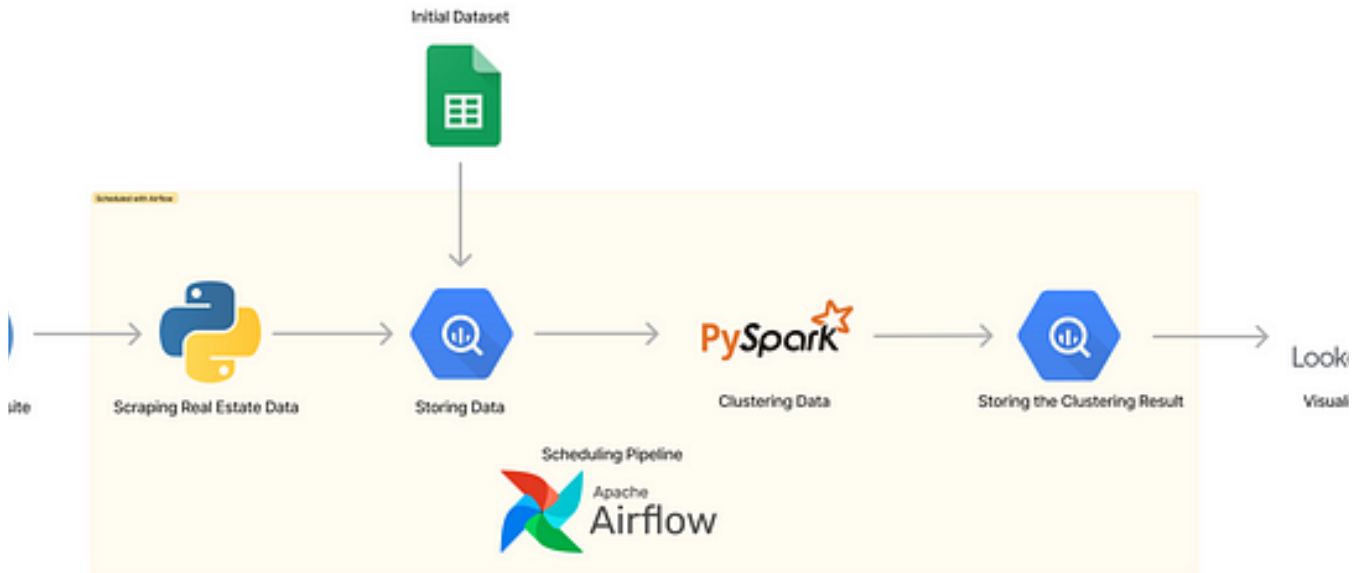
New_Reading_List

174 stories · 128 saves



Natural Language Processing

666 stories · 269 saves



Dana Fatadilla Rabba in Towards Dev

Building End-to-end Data Pipeline with Airflow, PySpark, and BigQuery

In today's data-driven world, extracting, transforming, and loading (ETL) processes play a vital role in handling and analyzing large...

4 min read · May 12



61



Apache Airflow



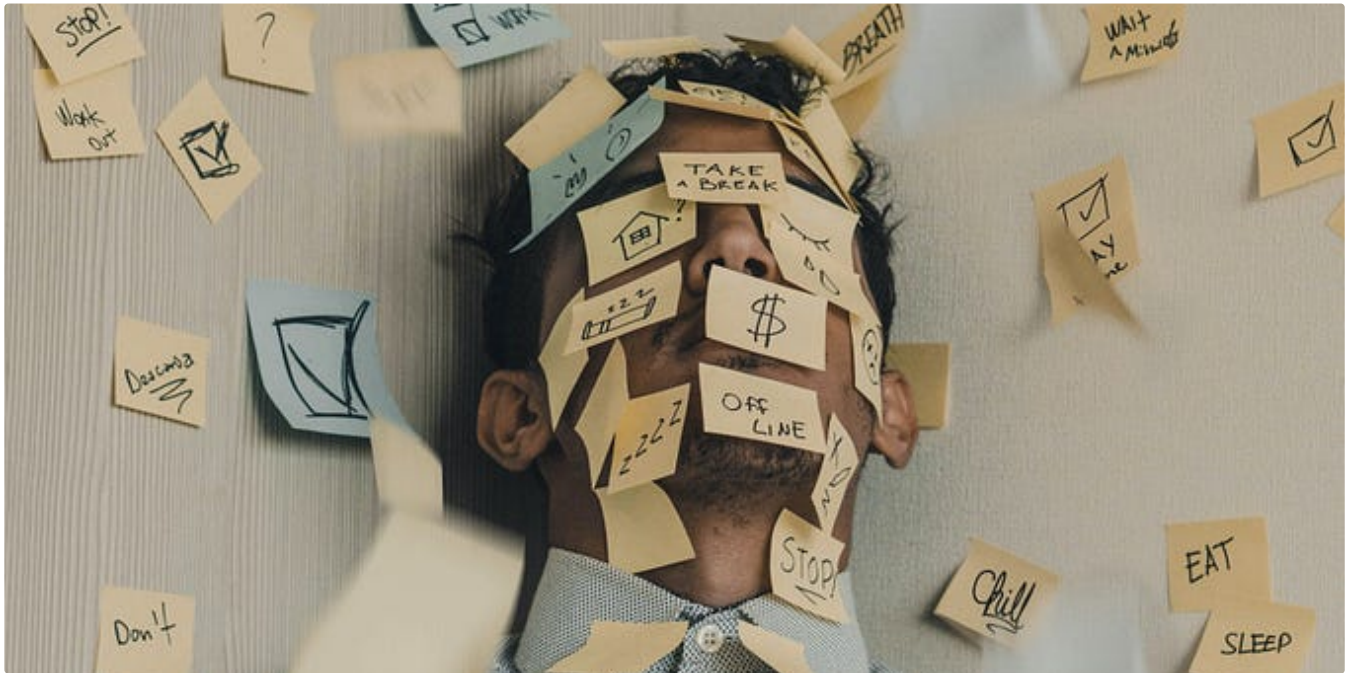
Akshay Thakare


How to Install Apache Airflow In Docker Container on EC2 Machine.

In this article, we'll cover how to install Airflow Docker container on ec2 machine in simple 3 steps.

4 min read · May 30

👏 1 💬



 Orchestra in Orchestra's Data Release Pipeline Blog

Why organisations fail data teams

Gartner reported in 2017 that upwards of 85% of data projects fail. Today, it seems the figure is similar for AI projects. These statistics...

6 min read · 5 days ago

👏 32 💬





Mohamadhasan Sarvandani

Learning Apache Airflow with simple examples

Apache Airflow is a powerful platform designed for workflow and data pipeline management (like the photo). It enables users to define...

6 min read · Jul 17



4



See more recommendations