



File_sensor_task

- Verifica o arquivo em intervalos regulares.
- Não monitora a pasta indefinidamente
- Não inicializa a DAG quando o arquivo for disponibilizado
- Não tem conhecimento das execuções anteriores da DAG



File_sensor_task

- **filepath:** verifica se o arquivo existe antes de prosseguir
- **fs_conn_id:** conexão com o arquivo através de conexão do Airflow. Conexão padrão fs_default



windturbine

- Gera um arquivo Json
 - ```
{"idtemp": "1", "powerfactor":
"0.8837929080361997",
"hydraulicpressure":
"78.86011124702158",
"temperature":
"25.279809506572597", "timestamp":
"2023-03-19 17:26:55.230351"}
```
- Vamos usar um arquivo pronto
- Notebook Python simula a geração do arquivo



```
import json
from random import uniform
import time
from datetime import datetime
id = 0
while True:
 id += 1
 dados_pf = uniform(0.7,1)
 #hydraulicpressure
 dados_hp = uniform(70,80)
 #temperature
 dados_tp = uniform(20,25)
 registro = {'idtemp' : str(id), 'powerfactor' : str(dados_pf), 'hydraulicpressure' : str(dados_hp) ,
 'temperature' : str(dados_tp) , 'timestamp' : str(datetime.now()) }
 #Data = json.dumps(registro)
 with open('C:/Users/usuario/Desktop/airflow/data/data.json', 'w') as fp:
 json.dump(registro, fp)
 print(dados_tp)
 time.sleep(5000)
```

📄 ↑ ↓ 🗑️ 5



# Notebook

# schedule\_interval

- A cada 3 minutos:  
    `"*/3 * * * *"`
- No desenvolvimento vamos usar  
    None



# PythonOperator

- Deverá ler o json
- Colocar as 5 variáveis no Xcom
- Excluir o arquivo



## BranchPythonOperator

- Se a temperatura for  $\geq 24$  graus manda Email de alerta
- Se não manda um email informativo





# PostgresOperator

- Cria a tabela
- Insere os dados



# Pré Etapas

- Criar conexão para file\_sensor\_task
- Criar variável com caminho do arquivo

