СЛЁРМ Обучение для инженеров и технических лидеров

slurm.io

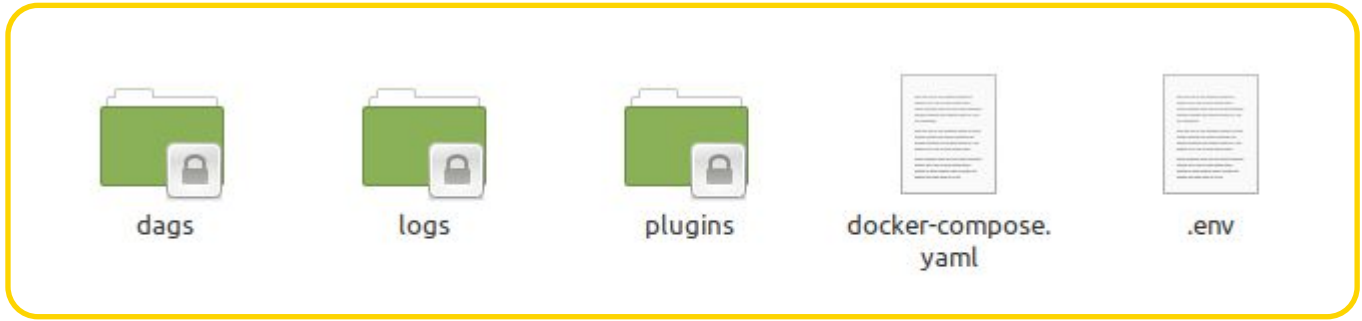
Дата-инженер

ETL-пайплайн в Apache Airflow

Ася Гайламазян

Первый пайплайн

Создание DAG

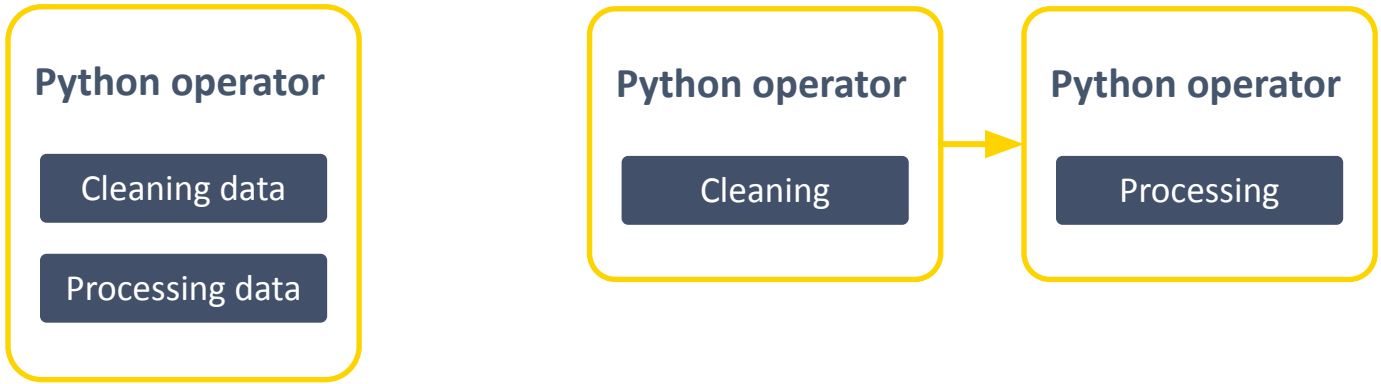


user_processing.py

```
from airflow import DAG
from datetime import datetime

with DAG('user_processing', start_date=datetime(2023,1,1),
schedule_interval='@daily', catchup=False) as dag:
    None
```

Tasks



Создание таблицы

```
from airflow import DAG
from airflow.providers.postgres.operators.postgres import PostgresOperator
from datetime import datetime

with DAG('user_processing', start_date=datetime(2023,1,1),
schedule_interval='@daily', catchup=False) as dag:
    create_table = PostgresOperator(task_id='create_table',
postgres_conn_id='postgres', sql='''CREATE TABLE IF NOT EXISTS users(firstname
TEXT NOT NULL, lastname TEXT NOT NULL, country TEXT NOT NULL, username TEXT NOT
NULL, password TEXT NOT NULL, email TEXT NOT NULL); ''')
```

Создание соединения

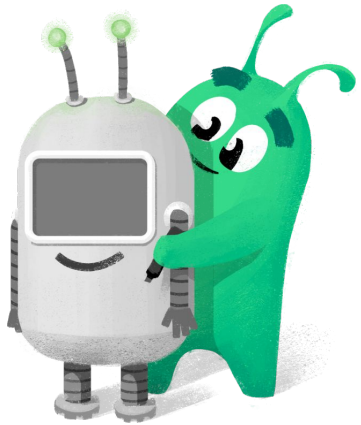
id postgres

type: Postgres

Host: postgres

login/password: airflow/airflow

port: 5432



Создание оператора сенсора

```
from airflow.providers.http.sensors.http import HttpSensor
…
is_api_available = HttpSensor(task_id='is_api_available',
http_conn_id='user_api', endpoint='api/')
```

Соединение с API

https://randomuser.me/

Извлечение данных из API (Extract)

```
from airflow.providers.http.operators.http import SimpleHttpOperator
import json
…
extract_user = SimpleHttpOperator(task_id='extract_user',
http_conn_id='user_api', endpoint='api/', method='GET',
response_filter=lambda response: json.loads(response.text))
```

Обработка юзеров. PythonOperator

```
from airflow.operators.python import PythonOperator
from pandas import json_normalize
…
process_user = PythonOperator(task_id='process_user',
python_callable=_process_user,)
```

//Функция _process_user объявляется отдельно,
вне менеджера контекста with

Функция обработки данных _process_user (Transform)

```
def _process_user(ti):
    user = ti.xcom_pull(task_ids="extract_user")
    user = user['results'][0]
    processed_user = json_normalize({ 'firstname':
user['name']['first'],'lastname': user['name']['last'], 'country':
user['location']['country'],'username': user['login']['username'],
        'password': user['login']['password'], 'email': user['email']})
    processed_user.to_csv('/tmp/processed_user.csv', index=None,
header=False)
```

Хранение пользователей (Load)

```
from airflow.providers.postgres.hooks.postgres import
PostgresHook
…
```

//Функция _store_user объявляется отдельно, вне менеджера контекста with

```
def _store_user():
    hook = PostgresHook(postgres_conn_id='postgres',)
    hook.copy_expert(sql="COPY users FROM stdin WITH DELIMITER
as ',' ", filename='/tmp/processed_user.csv')
```

//В менеджере контекста:
store_user = PythonOperator(task_id='store_user', python_callable=_store_user)

Оператор store_user и DAG

```
with DAG('user_processing', start_date=datetime(2023,1,1),
schedule_interval='@daily', catchup=False) as dag:
    create_table = PostgresOperator(task_id='create_table',
postgres_conn_id='postgres', sql='''
        CREATE TABLE IF NOT EXISTS
        users(firstname TEXT NOT NULL, lastname TEXT NOT NULL,  country TEXT NOT
NULL,
        username TEXT NOT NULL, password TEXT NOT NULL, email TEXT NOT NULL); ''')
    is_api_available = HttpSensor(task_id='is_api_available',
http_conn_id='user_api', endpoint='api/')
    exctract_user = SimpleHttpOperator(task_id='extract_user',
http_conn_id='user_api', endpoint='api/', method='GET', response_filter=lambda
response: json.loads(response.text))
    process_user = PythonOperator(task_id='process_user',
python_callable=_process_user,)
    store_user = PythonOperator(task_id='store_user', python_callable=_store_user)
```