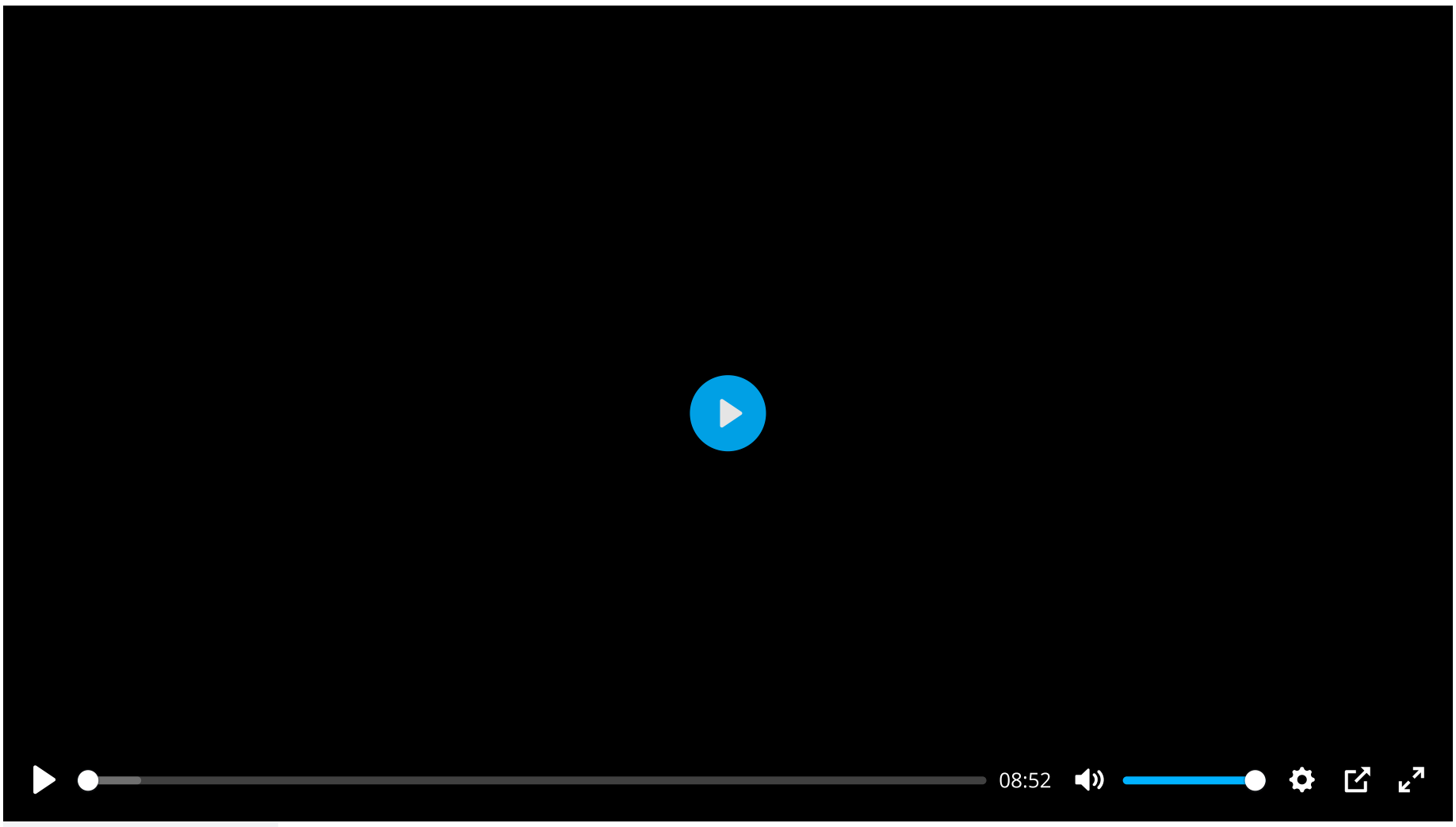
8.3 Создание и работа с DAG



Презентация к уроку 8.3.7

Текстовая расшифровка видео:

РАБОТА С PIPELINE. ЗАВЕРШЕНИЕ

План:

- Создание зависимостей в DAG;
- Создание промежуточной папки/tmp в воркере;
- Результат выполнения пайплайна;
- Отладка кода;
- Тестирование tasks и dags.

Создание зависимостей в DAG

Прежде мы видели в Web-UI, что таски следуют не друг за другом и выполняются в рандомном порядке, чтобы такого не было, в конец файла мы прописываем нужную последовательность с помощью оператора BitShift.

Вначале создается таблица, затем создается таск, в котором проверяется доступность API. Далее, извлекаем user'ы, обрабатываем их и сохраняем:

```
create_table >> is_api_available >> extract_user >> process_user >> store_user
```

Создание промежуточной папки/tmp в воркере

Мы можем проверить извлечение CSV-файла в промежуточную папку tmp, зайдя в контейнер с воркером:

```
sudo docker ps
sudo docker exec -it airflow_airflow-worker_1 /bin/bash
airflow@007c8663c83a:/opt/airflow$ ls /tmp
```

Результат выполнения пайплайна

Также мы сможем проверить успешное выполнение всего пайплайна, когда посмотрим в таблицу «Postgres» и в таблицу «Users»:

```
sudo docker exec -it airflow_postgres_1 psql -U airflow
airflow=# SELECT * FROM users;
```

Отладка кода

Мы можем сделать отладку таска, отдельных тасков и всего DAG в контейнере «Scheduler». В нем находятся процессоры Airflow и по нему можно увидеть Help по команде «Airflow -h»:

```
docker exec -it имя_scheduler /bin/bash
airflow -h
```

Тестирование tasks и dags

Примеры тестов:

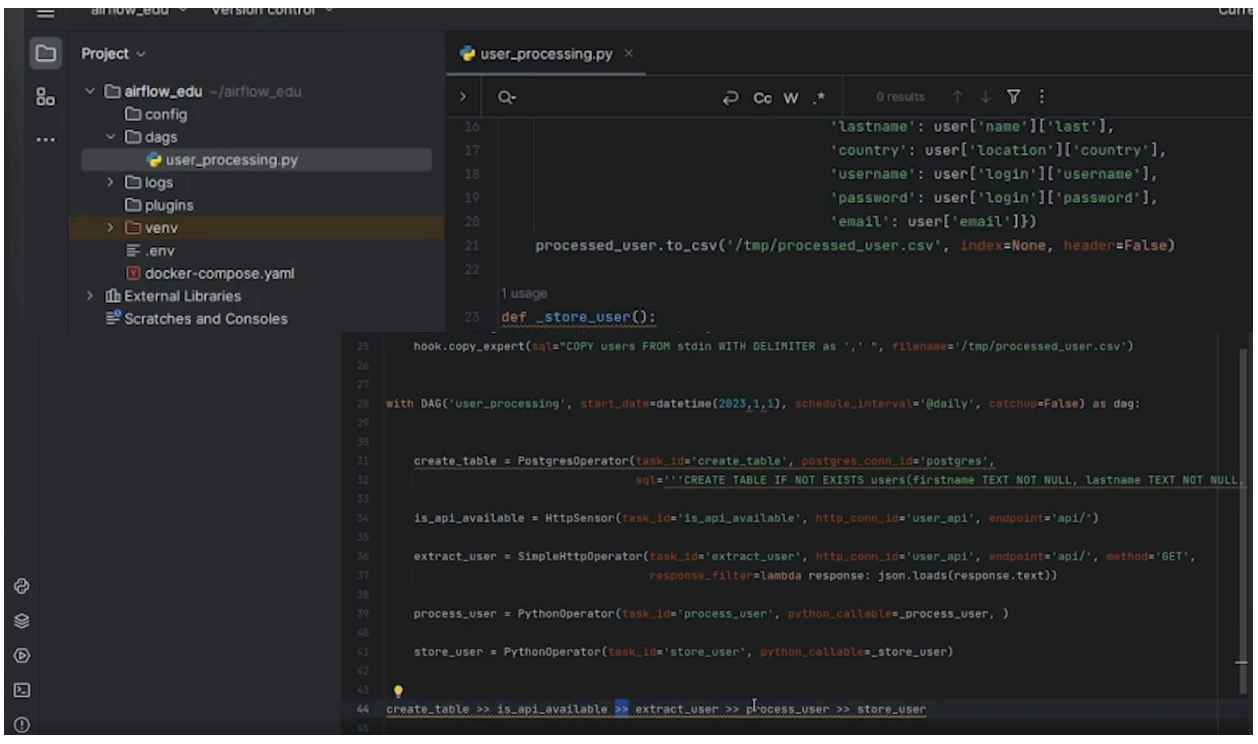
```
airflow dags test user_processing 2023-01-01
```

Для отдельных тасков:

```
airflow tasks test user_processing create_table 2023-01-01
```

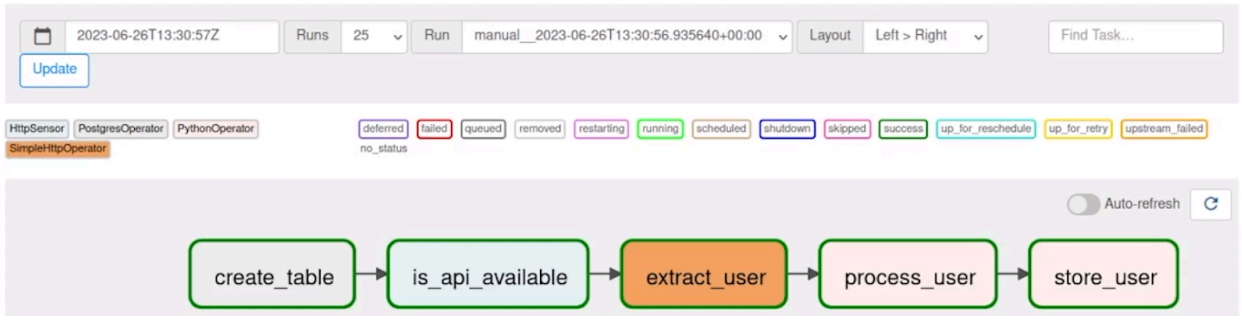
Рассмотрим пример:

Мы видим DAG без связности:



Добавим строчку в конец Python-файла для того, чтобы создать связность. Добавляем без отступа, так как это находится вне менеджера контекста по работе с DAG.

Обновляем страницу и видим, что появились стрелки, обозначающие связность:



Проверим наличие файлов в папке tmp в контейнере «Worker». Видим наличие processed user.csv:

```
asya@asya-Aspire-XC-886:~/airflow_edu$ docker exec -it airflow_edu-airflow-worker-1 /bin/bash
airflow@7b8c5176d225:/opt/airflow$ cd /tmp
airflow@7b8c5176d225:/tmp$ ls
processed_user.csv  pymp-uvmjfdte  tmpqbmxmg08  tmpt5vwwj_2
airflow@7b8c5176d225:/tmp$
```

В содержимом хранится строка с данными user’a, полученная из API.

Проверим, появились ли данные в таблице. Для этого зайдем в контейнер «Postgres» и используем команду «**psql**», которая сразу подключит к базе, находящейся внутри контейнера. **User и пароль есть в docker-compose-файле в разделе с описанием базы.** Смотрим:

```
asya@asya-Aspire-XC-886:~/airflow_edu$ docker exec -it airflow_edu-airflow-worker-1 /bin/bash
airflow@7b8c5176d225:/opt/airflow$ cd /tmp
airflow@7b8c5176d225:/tmp$ ls
processed_user.csv  pymp-uvmjfdte  tmpqbmxmg08  tmpt5vwwj_2
airflow@7b8c5176d225:/tmp$ cat processed_user.csv
Jesper,Wilson,Norway,happytiger782,fdsa,jesper.wilson@example.com
airflow@7b8c5176d225:/tmp$ exit
exit
asya@asya-Aspire-XC-886:~/airflow_edu$ docker exec -it airflow_edu-postgres-1 psql -U airflow
psql (13.11 (Debian 13.11-1.pgdg120+1))
Type "help" for help.

airflow=# SELECT * FROM user;
  user
---------
 airflow
(1 row)

airflow=# SELECT * FROM users;
  firstname  |  lastname   | country |     username     | password |               email
-------------+-------------+---------+------------------+----------+-----------------------------------
 Tverdislava | Zhuravlenko | Ukraine | happydog172      | melinda  | tverdislava.zhuravlenko@example.com
 Jesper      | Wilson      | Norway  | happytiger782    | fdsa     | jesper.wilson@example.com
 Fatma       | Aclan       | Turkey  | yellowleopard161 | united   | fatma.aclan@example.com
(3 rows)
```

Появились строчки по каждому успешному выполнению вставки.

Попробуем протестировать таски и DAG. Для этого заходим в контейнер «Scheduler», открываем терминал и видим, что команда «Airflow help» показывает, что мы можем сделать процессы на Airflow. Здесь есть Manage tasks и Manage DAG’s. Также можно производить тесты. Вводим айдишник DAG и обязательно указываем execution date:

```
Optional Arguments:
  -h, --help         show this help message and exit
airflow@a2d236165387:/opt/airflow$ airflow dags test user_processing 2023-01-01
```

Мы видим статус «success»:

```
[2023-06-26 15:27:03,400] {python.py:183} INFO - Done. Returned value was: None
[2023-06-26T15:27:03.400+0000] {python.py:183} INFO - Done. Returned value was: None
[2023-06-26 15:27:03,403] {taskinstance.py:1350} INFO - Marking task as SUCCESS. dag_id=user_processing,
 execution_date=20230101T000000, start_date=, end_date=20230626T152703
[2023-06-26T15:27:03.403+0000] {taskinstance.py:1350} INFO - Marking task as SUCCESS. dag_id=user_proces
user, execution_date=20230101T000000, start_date=, end_date=20230626T152703
[2023-06-26T15:27:03.419+0000] {dag.py:3691} INFO - process_user ran successfully!
[2023-06-26T15:27:03.419+0000] {dag.py:3694} INFO - ********************************************************
[2023-06-26T15:27:03.426+0000] {dag.py:3683} INFO - ********************************************************
[2023-06-26T15:27:03.426+0000] {dag.py:3687} INFO - Running task store_user
[2023-06-26 15:27:03,461] {taskinstance.py:1547} INFO - Exporting env vars: AIRFLOW_CTX_DAG_OWNER='airfl
='user_processing' AIRFLOW_CTX_TASK_ID='store_user' AIRFLOW_CTX_EXECUTION_DATE='2023-01-01T00:00:00+00:00
BER='1' AIRFLOW_CTX_DAG_RUN_ID='manual__2023-01-01T00:00:00+00:00'
[2023-06-26T15:27:03.461+0000] {taskinstance.py:1547} INFO - Exporting env vars: AIRFLOW_CTX_DAG_OWNER='a
AG_ID='user_processing' AIRFLOW_CTX_TASK_ID='store_user' AIRFLOW_CTX_EXECUTION_DATE='2023-01-01T00:00:00+
Y_NUMBER='1' AIRFLOW_CTX_DAG_RUN_ID='manual__2023-01-01T00:00:00+00:00'
[2023-06-26T15:27:03.461+0000] {postgres.py:159} INFO - Running copy expert: COPY users FROM stdin WITH D
[2023-06-26T15:27:03.461+0000] {taskinstance.py:1547} INFO - Exporting env vars: AIRFLOW_CTX_DAG_OWNER='airflow' AIRFLOW_CTX_D
AG_ID='user_processing' AIRFLOW_CTX_TASK_ID='store_user' AIRFLOW_CTX_EXECUTION_DATE='2023-01-01T00:00:00+00:00' AIRFLOW_CTX_TR
Y_NUMBER='1' AIRFLOW_CTX_DAG_RUN_ID='manual__2023-01-01T00:00:00+00:00'
[2023-06-26T15:27:03.461+0000] {postgres.py:159} INFO - Running copy expert: COPY users FROM stdin WITH DELIMITER as ',' , fil
ename: /tmp/processed_user.csv
[2023-06-26T15:27:03.463+0000] {base.py:73} INFO - Using connection ID 'postgres' for task execution.
[2023-06-26 15:27:03,469] {python.py:183} INFO - Done. Returned value was: None
[2023-06-26T15:27:03.469+0000] {python.py:183} INFO - Done. Returned value was: None
[2023-06-26 15:27:03,473] {taskinstance.py:1350} INFO - Marking task as SUCCESS. dag_id=user_processing, task_id=store_user, e
xecution_date=20230101T000000, start_date=, end_date=20230626T152703
[2023-06-26T15:27:03.473+0000] {taskinstance.py:1350} INFO - Marking task as SUCCESS. dag_id=user_processing, task_id=store_us
er, execution_date=20230101T000000, start_date=, end_date=20230626T152703
[2023-06-26T15:27:03.486+0000] {dag.py:3691} INFO - store_user ran successfully!
[2023-06-26T15:27:03.486+0000] {dag.py:3694} INFO - ********************************************************
[2023-06-26T15:27:03.492+0000] {dagrun.py:616} INFO - Marking run <DagRun user_processing @ 2023-01-01T00:00:00+00:00: manual_
_2023-01-01T00:00:00+00:00, state:running, queued_at: None. externally triggered: False> successful
[2023-06-26T15:27:03.493+0000] {dagrun.py:682} INFO - DagRun Finished: dag_id=user_processing, execution_date=2023-01-01T00:00
:00+00:00, run_id=manual__2023-01-01T00:00:00+00:00, run_start_date=2023-01-01T00:00:00+00:00, run_end_date=2023-06-26 15:27:0
3.492999+00:00, run_duration=15262023.492999, state=success, external_trigger=False, run_type=manual, data_interval_start=2023
-01-01T00:00:00+00:00, data_interval_end=2023-01-02T00:00:00+00:00, dag_hash=None
airflow@a2d236165387:/opt/airflow$
```

Проверим, появились ли данные в таблице «Postgres»:

```
asya@asya-Aspire-XC-886:~/airflow_edu$ docker exec -it  airflow_edu-postgres-1 psql -U airflow
psql (13.11 (Debian 13.11-1.pgdg120+1))
Type "help" for help.

airflow=# SELECT * FROM users;
  firstname  |  lastname   | country |     username     | password |               email
-------------+-------------+---------+------------------+----------+-----------------------------------
 Tverdislava | Zhuravlenko | Ukraine | happydog172      | melinda  | tverdislava.zhuravlenko@example.com
 Jesper      | Wilson      | Norway  | happytiger782    | fdsa     | jesper.wilson@example.com
 Fatma       | Aclan       | Turkey  | yellowleopard161 | united   | fatma.aclan@example.com
 Diether     | Hansmann    | Germany | whitekoala218    | smoking  | diether.hansmann@example.com
(4 rows)

airflow=# \q
asya@asya-Aspire-XC-886:~/airflow_edu$
```

У нас было три строчки, теперь появилась четвертая.

Выполнение тестов тоже добавляет данные в таблицу.

Протестируем отдельный таск. В качестве примера возьмем таск проверки доступности API. Здесь важно отметить, что сенсоры делают «**Poking: api**», то есть проверяют доступность. Если критерии успешны, то они продолжают выполнение:

```
airflow command error: argument GROUP_OR_COMMAND: invalid choice: 'task' (choose from 'celery', 'cheat-sheet', 'config', 'conn
ections', 'dag-processor', 'dags', 'db', 'info', 'jobs', 'kerberos', 'kubernetes', 'plugins', 'pools', 'providers', 'roles', '
rotate-fernet-key', 'scheduler', 'standalone', 'sync-perm', 'tasks', 'triggerer', 'users', 'variables', 'version', 'webserver'
), see help above.
airflow@a2d236165387:/opt/airflow$ airflow tasks test user_processing is_api_available 2023-01-01
[2023-06-26T15:28:36.518+0000] {dagbag.py:541} INFO - Filling up the DagBag from /opt/***/dags
[2023-06-26T15:28:38.698+0000] {taskinstance.py:1103} INFO - Dependencies all met for dep_context=non-requeueable deps ti=<Tas
kInstance: user_processing.is_api_available manual__2023-01-01T00:00:00+00:00 [success]>
[2023-06-26T15:28:38.707+0000] {taskinstance.py:1103} INFO - Dependencies all met for dep_context=requeueable deps ti=<TaskIns
tance: user_processing.is_api_available manual__2023-01-01T00:00:00+00:00 [success]>
[2023-06-26T15:28:38.708+0000] {taskinstance.py:1308} INFO - Starting attempt 1 of 1
[2023-06-26T15:28:38.718+0000] {taskinstance.py:1327} INFO - Executing <Task(HttpSensor): is_api_available> on 2023-01-01 00:0
0:00+00:00
[2023-06-26T15:28:38.884+0000] {taskinstance.py:1547} INFO - Exporting env vars: AIRFLOW_CTX_DAG_OWNER='***' AIRFLOW_CTX_DAG_I
D='user_processing' AIRFLOW_CTX_TASK_ID='is_api_available' AIRFLOW_CTX_EXECUTION_DATE='2023-01-01T00:00:00+00:00' AIRFLOW_CTX_
TRY_NUMBER='1' AIRFLOW_CTX_DAG_RUN_ID='manual__2023-01-01T00:00:00+00:00'
[2023-06-26T15:28:38.885+0000] {http.py:122} INFO - Poking: api/
[2023-06-26T15:28:38.889+0000] {base.py:73} INFO - Using connection ID 'user_api' for task execution.
[2023-06-26T15:28:39.403+0000] {base.py:255} INFO - Success criteria met. Exiting.
[2023-06-26T15:28:39.405+0000] {taskinstance.py:1350} INFO - Marking task as SUCCESS. dag_id=user_processing, task_id=is_api_a
vailable, execution_date=20230101T000000, start_date=, end_date=20230626T152839
airflow@a2d236165387:/opt/airflow$
```

Мы завершили наш первый пайплайн и работу над созданием DAG. Мы посмотрели на то, как правильно отлаживать DAG, как тестировать отдельные таски и весь DAG целиком, а также узнали, что результат записывается туда, куда и при выполнении через Web-UI.