ELT

ETL - метод. обработки

DWH - бд все структурированно, запрос быстро раб.

DAte lake - в сыром виде, глубоко копнуть можно

Extract, Transform, Load
↓ Выгрузка ↓ Преобразование ↓ Загрузка в БД

отправка данных на почту (отправка результата)

DAG - контейнер для задач.
- это набор задач, которые выполняются в какой-то последовательности от начала и до конца.
↓ является Class.

TASK - это экземпляры задач.

которые создаются с помощью операторов.
↓
операторы нужны для выполнения действий с данными

Рекомендации писать ~~DAG~~ ETL

1. Весь код декларативный
↓ = 2 + 2 - знаем результат

императивный код
↓ x + y = x3

2. Идемпотентность

x+0=0 0+1=1

Это при повторном процессе операции будет 1 табл, а не 2.

+ 3 есть необязательные аргументы DAG

clear-уд.

Operator - основ. компоненты, из которых состоит DAG
- нужны для выполнения действий PythonOperator, EmailOperator

↓ виды
- Трансферы - нужны передать данные из одного источника в другой
- sensor - проверяет наличие чего-либо, где-либо
  есть ли такой файл.
- Action - выполнять конкр. функц. (запуск python, отправка на почту)

Зависимости
1. Параллельные $t1 \gg [t2, t3]$ ① → ②, ③
2. Последовательные $t1 \gg t2$ ① → ②

stepic
3.592 секунд
01.02.24

vrorkч24@gmail.com
9870vwork

Task-I - так называемый оператор, который позволяет выполнять ту или иную операцию

Hooks - позволяют взаимод. с внешними подключениями, например БД postgres

1 docker-compose down -v

2 examples меняем на false

3 docker-compose up airflow-init

4. docker-compose ~~down~~ up -d

Из чего состоит AirFlow

Сканирует

Scheduler - рекурсивно сканирует папку с дагами, когда нужно запускать

↓

после того как решит, что пора запускать в игру входит

Запуск задач

Executor - отвечает за запуск задач, и они отправляются в IPC очередь

↓ после включается

Вытаскивает задачи из очереди

Worker - он вытаскивает задачу из очереди и начинает исполнение

↘ для Операторов Sensor возможно ситуация повторения задачи, которая именуется как Rescheduler, это постановка обратно в очередь пока не достигнем нужного события.

↓

Status задачи Success - выполнено

Failed - задача падает или повтор.

2 типа

1 ПК local — 2+ ПК

sequential Executor - использует sqlite бд. не умеет запускать параллельно, только

Local Executor - требует поставить минимум Postgres

Celery Executor - реализовать кластер

kubernetes Executor - позволяет поднять AirFlow в кубер кластере - то есть задачи будут каждая в своем собственном контейнере

-1 по умолч.

Pools - это механизм ограничения параллелизма задач. Если очень много будет задач, сервак упадет. поэтому нужно ограничение. Pools - 10 и будет max (10) задач

Контекст

- это информация доступная во время выполнения Operator
она помогает работать задачам вместе друг с другом

Контекст содержит след. инфу.

1. Execution Date - дата выполнения (не дата запуска)
2. Task Instance - экземпляр задачи
3. Parameters - параметр - доп параметры
4. Шаблон - Templates - это перемен или знач. которые можно использ. в ходе задачи с помощью выражения jinja

Jinja - это шаблонизатор, это код напрямую в текст

Макросы - спец. команда, вида {{ ds }} - для упрощ. выполн. задач.

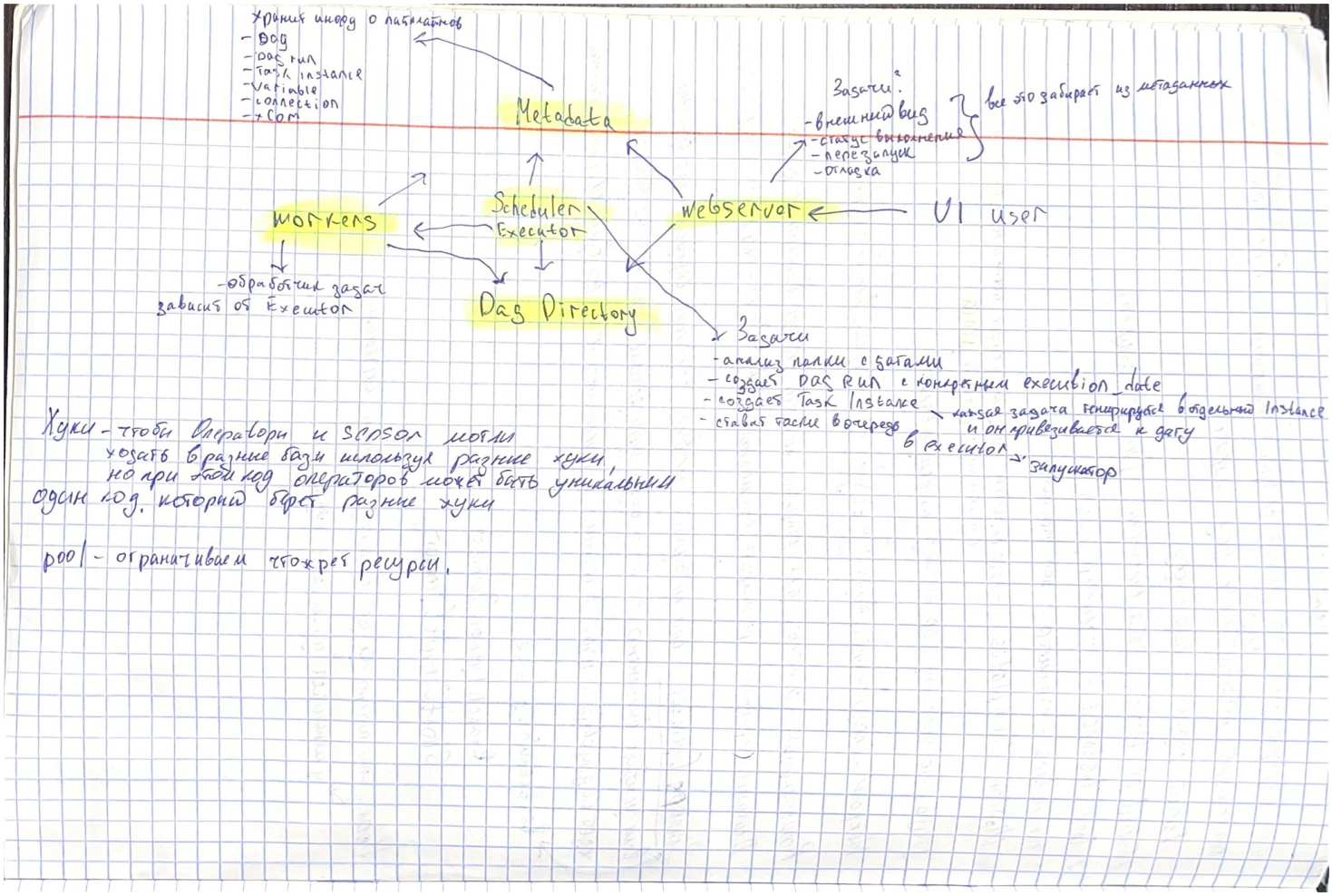
Нотификация - использ. для отправки на почту о статусе выполнения задач

Плагины - позволяет расширить возможности Airflow + операторы, сенсоры и тд.

Sensors Operator - они проверяют условие в течение определенного времени
Если условие выполнено, то DAG будет идти дальше, если нет то sensor
ждет след. проверки и так в течении определенного кол.-во времени, или бесконечно

ветвления: if else
- BranchPythonOperator
- ShortCircuitOperator

Hook - это интерфейс для взаимодействия с внеш. системами
↘ уменьшает дубликаты кода



Хуки - чтобы операторы и sensor могли ходить в разные базы используя разные хуки, но при этом код операторов может быть уникальным один код, который берет разные хуки

pool - ограничиваем что-то ресурсы.

Хуки, операторы, сенсоры

Connections - для подключения к различным базам

Хук - это интерфейс для соединения - PostgresHook

Операторы - шаблоны для тасков - PostgresOperator

Сенсоры - ожидают момента наступления какого-либо события

↓ параметры → ExternalTaskSensor

timeout
soft_fail
poke_interval
mode - poke, reschedule

Ветвление

BranchPythonOperator

Шаблоны Jinja

{{ ds }}

...

Макросы ↑ - для упрощения задач.

xcom - механизм позволяющий таскам взаимодействовать друг с другом

↓ методы

↓

xcom_push - передает аргументы, которые мы хотим получить в другой таске

xcom_pull - забирает эти параметры

TaskFlow API

↓ каждая таска взаимодействует друг с другом напрямую, т.е. результат работы одного таска является входными параметрами для другого таска.

~~SubDags~~ - дочерние даги

↓ переиспользование кода

~~визуальная группировка~~

TaskGroup - цели одинаковые

- группировка задач
- переиспользование кода

Динамическое создание дагов