

# APACHE AIRFLOW

КАРТАШОВ АНДРЕЙ

[apk92@icloud.com](mailto:apk92@icloud.com)

© ООО «Учебный центр «Коммерсант», 2023



# Введение в Apache Airflow

МОДУЛЬ 1



ШКОЛА БОЛЬШИХ ДАННЫХ

# Программа курса

- ❖ Введение в Airflow
- ❖ Основные абстракции
- ❖ Написание простых DAG
- ❖ Дополнительные возможности
- ❖ Расширенные возможности
- ❖ Тонкости конфигурирования и best/bad practice
- ❖ Введение в основные инструменты DE
- ❖ Подготовка данных

# Что такое Apache Airflow?

- ❖ **Airflow** - cron на максималках (если совсем просто)
- ❖ **Airflow** - написан на Python
- ❖ **Airflow** - дешево, удобно, быстро
- ❖ **Airflow** - open source с огромным комьюнити

**Airflow** is a platform created by the community to programmatically author, schedule and monitor workflows.



# Почему Airflow?

Масштабируемость  
модульная архитектура,  
очередь сообщений

Кастомизация  
возможность настройки  
собственных операторов

Инструментарий  
WEB UI, CLI, REST API

Интеграция со множеством  
сервисов и источников  
данных  
Apache [Spark, Hive, Hadoop],  
MySQL, Postgres, MongoDB,  
Redis e.t.c.

Мониторинг и алерting  
поддерживается множество  
интеграций с различными  
сервисами сбора и отправки  
логов и метрик

Ролевой доступ  
5 ролей по дефолту +  
возможна интеграция с  
Active Directory

Поддержка и тестирование  
Можно добавлять базовые  
unit тесты, которые могут  
проверять как пайплайны в  
целом, так и отдельные задачи



# История создания Airflow

Apache Airflow **впервые появился в 2014** году в компании Airbnb.

**С января 2019** года является Top-Level проектом в организации Apache Foundation.

Среди пользователей этого инструмента есть такие компании как **Adobe, Lyft, Uber, Reddit, Google, Rambler, Alfa-Bank, Beeline, Sber** и многие другие не менее известные компании.

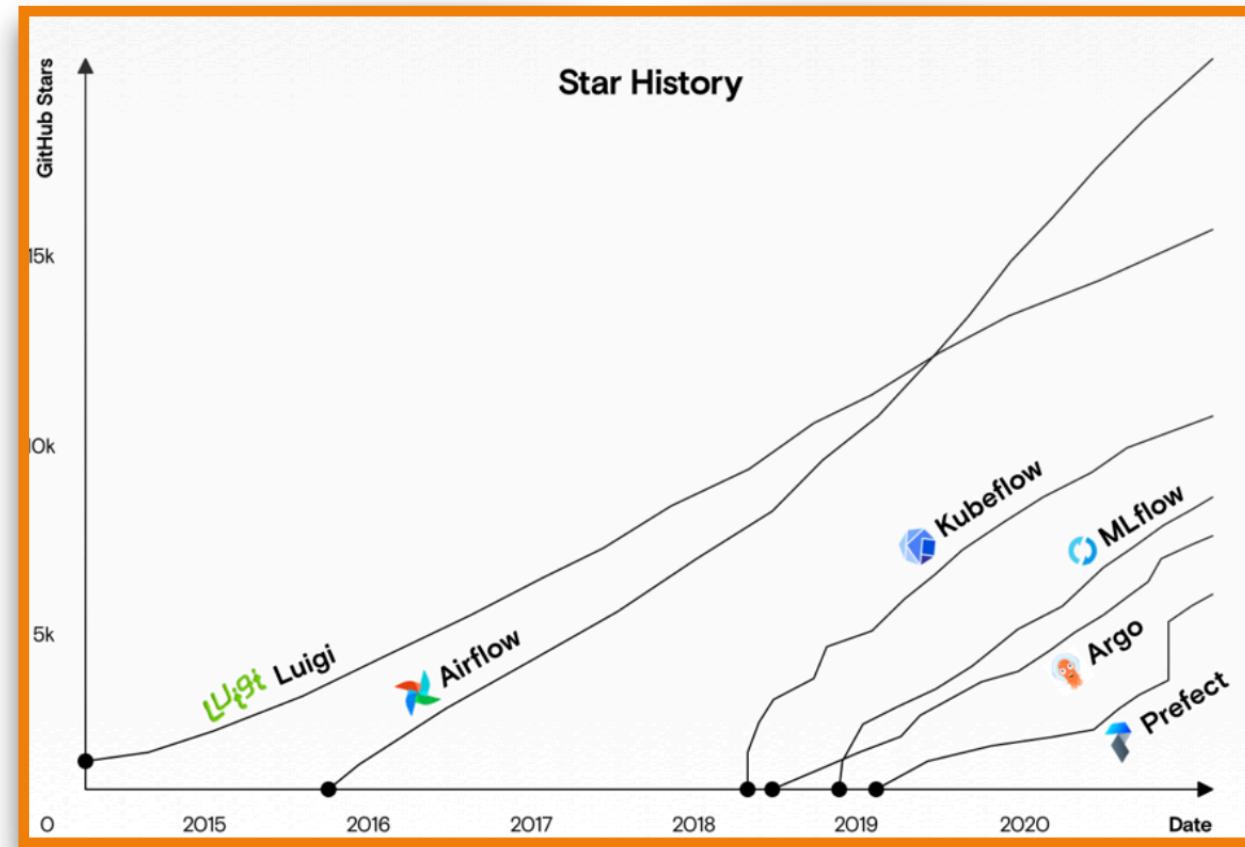
В декабре 2020 года вышла версия 2.0 в которой появилось очень много серьёзных улучшений.



ШКОЛА БОЛЬШИХ ДАННЫХ

# Аналоги и конкуренты

- ❖ Apache Oozie
- ❖ Apache NiFi
- ❖ Luigi (Spotify)
- ❖ Azkaban (LinkedIn)
- ❖ Perfect
- ❖ Cron



# Сравнение с конкурентами



- + Python для DAG's
- + Лучший WEB UI и API
- + Продвинутые метрики
- + Возможность создания сложных workflow's
- + Connections to HDFS, HIVE и т.д.
- + Будущее + комьюнити



- Java or XML для DAG's
- ужасный WEB UI и Java API
- умирающий, маленькое комьюнити



ШКОЛА БОЛЬШИХ ДАННЫХ

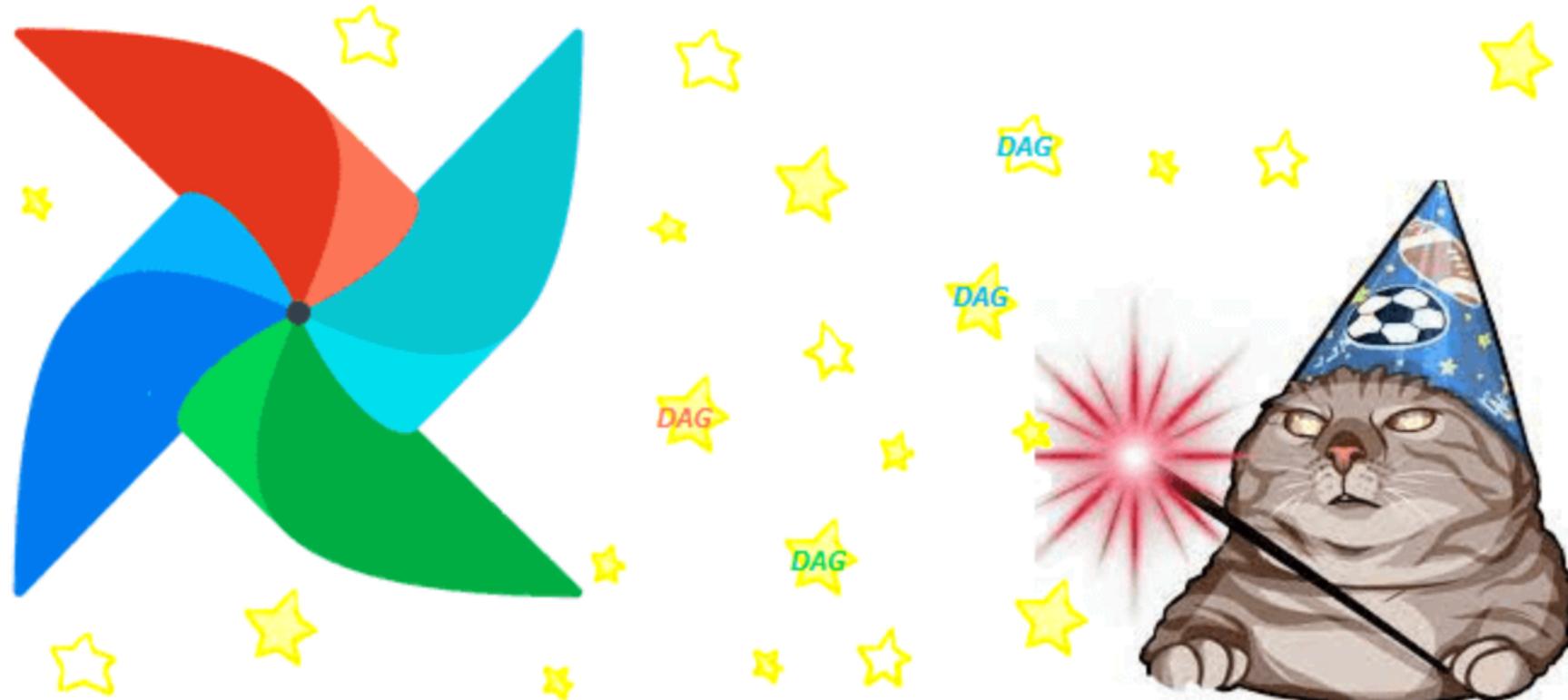
# Оркестрация

- ❖ Airflow - инструмент оркестрации
- ❖ Airflow - НЕ инструмент разработки
  - ✓ Разработка и отладка проходит отдельно
  - ✓ Проект «заворачивается» в Airflow
- ❖ Airflow - НЕ ETL инструмент
- ❖ «Синдром молотка»
- ❖ BigData

# Где применяется?

- ❖ Создание конвейеров обработки больших данных
  - ✓ Оркестрация Spark приложений
- ❖ Замена cron
- ❖ Построение кластерной инфраструктуры обработки больших данных
  - ✓ При наличии Hadoop
- ❖ Использование **НЕ** data инженерами
- ❖ Разработка собственных операторов

# Вопросы?

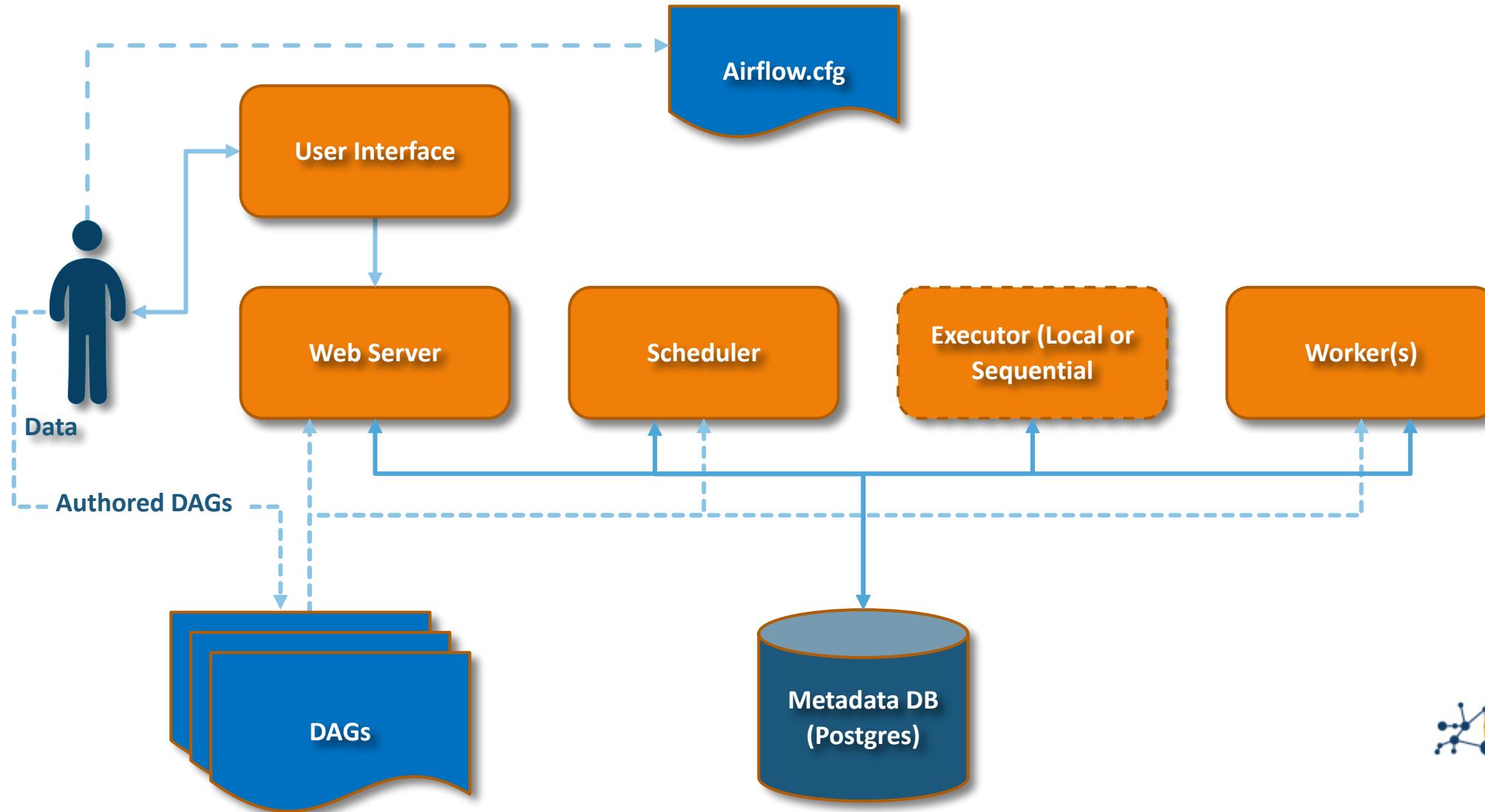




# Основные абстракции и компоненты

МОДУЛЬ 2

# Верхнеуровневая архитектура



# Программные компоненты

Webserver

Worker

Scheduler

CLI



ШКОЛА БОЛЬШИХ ДАННЫХ

# Абстракции: DAG

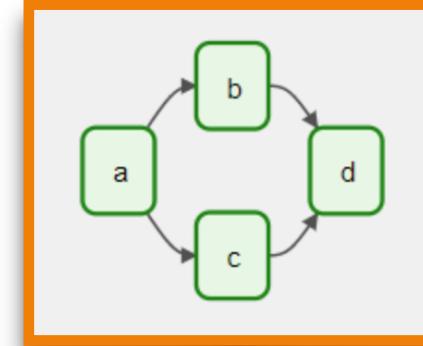
Набор инструкций для исполнения:

- ❖ Python file
  - ❖ Содержит композицию операторов
    - ❖ start\_date
  - ❖ Schedule

**Best practice:**  
один python file == один DAG

A **DAG** (Directed Acyclic Graph) is the core concept of Airflow, collecting Tasks together, organized with dependencies and relationships to say how they should run.

Basic example DAG



ШКОЛА БОЛЬШИХ ДАННЫХ

# Абстракции: operator

While **DAGs** describe how to run a workflow,  
**operators** determine what actually gets done

- ❖ Минимальная единица исполняемого кода
- ❖ Операторы могут использоваться на разных узлах
- ❖ Включаются в DAG
- ❖ Параметризирован

# Абстракции: operator

## Основные типы и виды операторов

### Назначение

- ❖ **Operator:** базовый тип - действие
- ❖ **Sensor:** ожидание события
- ❖ **Transfer:** перенос информации

### «Язык»

- ❖ PythonOperator
- ❖ BashOperator
- ❖ EmailOperator
- ❖ SimpleHttpOperator
- ❖ PostgresOperator
- ❖ ...

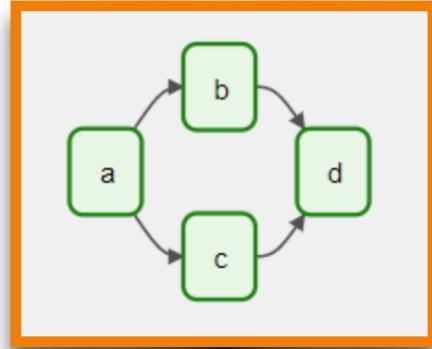


ШКОЛА БОЛЬШИХ ДАННЫХ

# Абстракции: композиция операторов

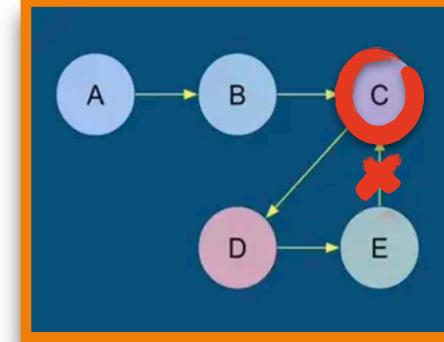
operator2 **следует** за operator1 (operator1  $\gg$  operator2)

Basic example DAG



a  $\gg$  [b, c]  $\gg$  d

! DAG Import Errors (1)



Broken DAG: [/opt/airflow/dags/broken\_dag.py]

Cycle detected in DAG: broken\_dag. Faulty task: c



ШКОЛА БОЛЬШИХ ДАННЫХ

# Абстракции: Variable

- ❖ пара «key - value»
- ❖ хранятся в бд airflow
- ❖ операции (get, delete, set, update ...)

#setInWebUI

Admin -> Variables -> +

#import

from airflow.models import Variable

#use

foo = Variable.get('foo')

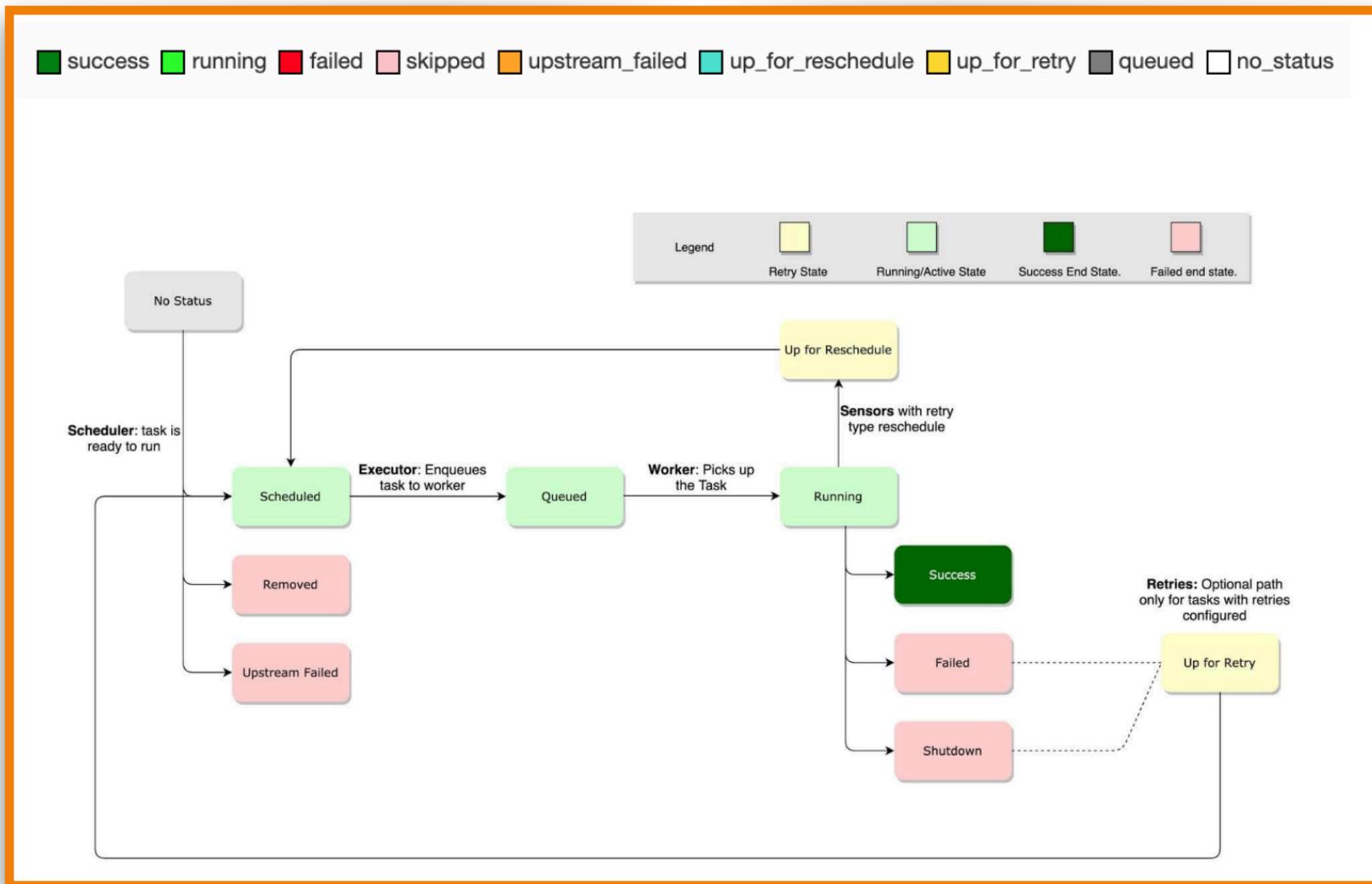


ШКОЛА БОЛЬШИХ ДАННЫХ

# Исполнение и dag run

- ❖ **dag run:** DAG на этапе исполнения
  - ✓ execution date
  - HE процесс
  
- ❖ **task:** оператор на этапе исполнения
  - процесс

# Task States



# Обзор WEB UI

## Возможности:

- ❖ Просмотр состояния и истории DAGов
- ❖ Статистика выполнения DAGов
- ❖ Логи
- ❖ Рендер переменных
- ❖ Просмотр исходного кода DAGa
- ❖ Различное конфигурирование  
(переменные, коннекты...)
- ❖ Старт - стоп DAGов
- ❖ Визуальное представление DAG
- ❖ И многое другое

# Вопросы?



ШКОЛА БОЛЬШИХ ДАННЫХ

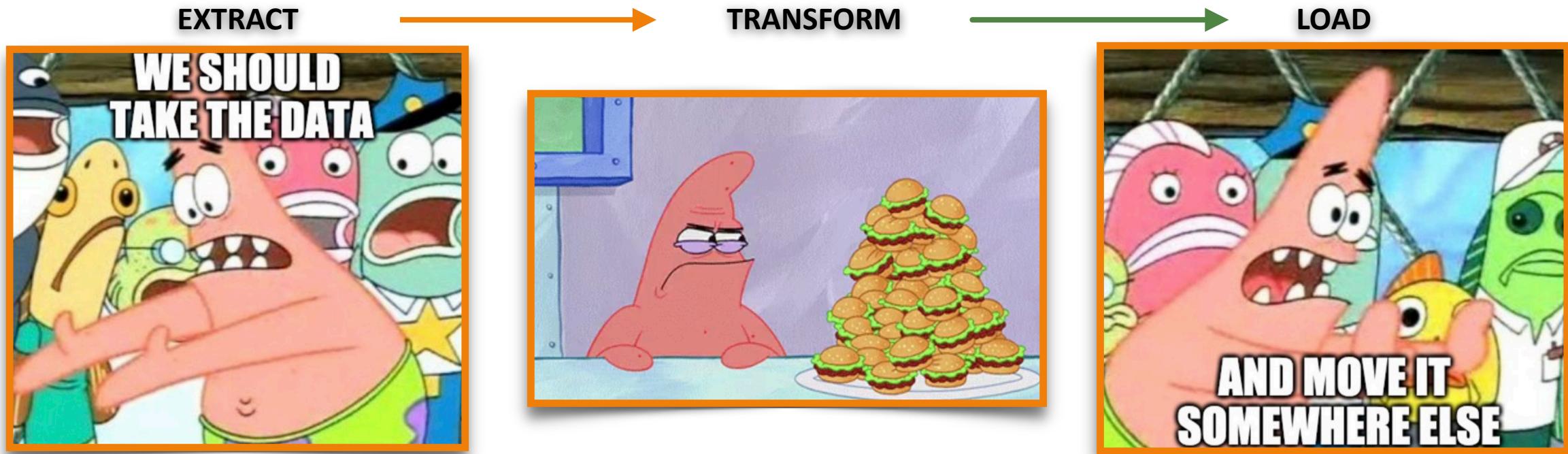


# Разработка конвейеров данных

МОДУЛЬ 3



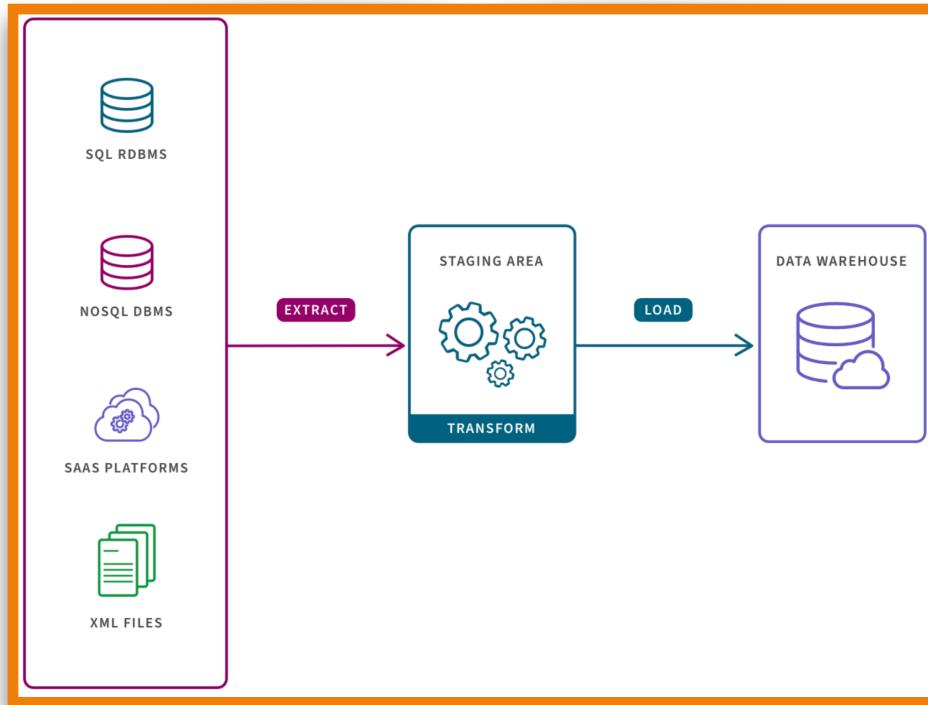
# ETL



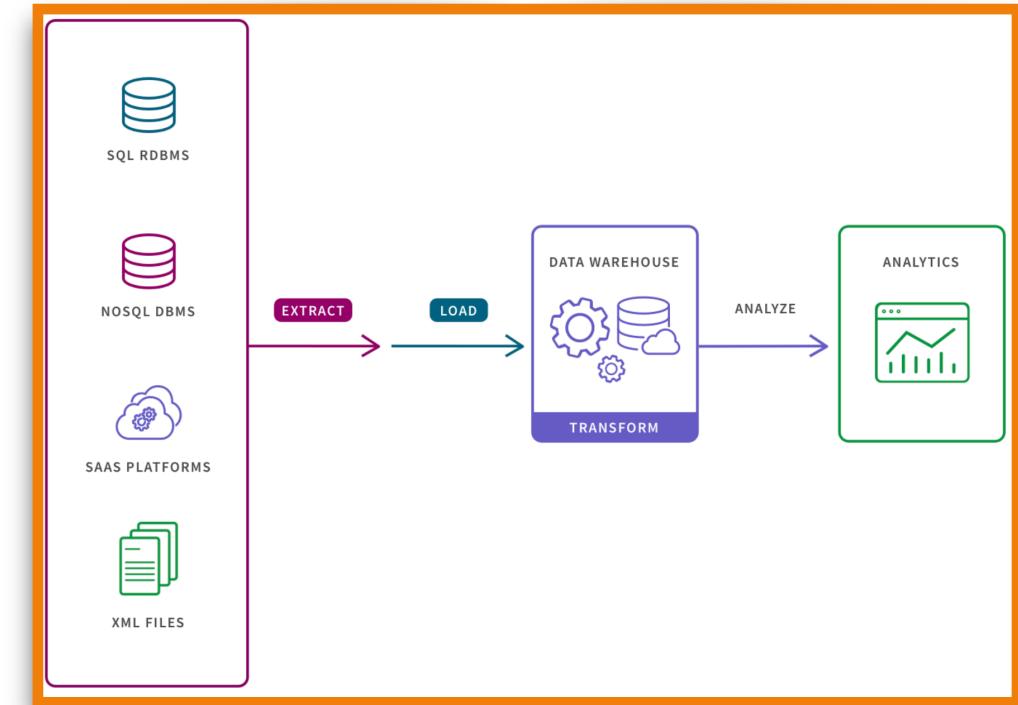
ШКОЛА БОЛЬШИХ ДАННЫХ

# Data Pipelines (ELT) vs ETL Pipelines

ETL



ELT



ШКОЛА БОЛЬШИХ ДАННЫХ

# dataflow vs workflow

## dataflow

- ❖ поток данных
- ❖ обеспечивается хранение данных на всем пути

## workflow

- ❖ поток «управления»
- ❖ нет привязки к данным



ШКОЛА БОЛЬШИХ ДАННЫХ

# Pipeline и Airflow

- ❖ **DAG == pipeline**  
✓ operator == этап конвейера
- ❖ **workflow (!= dataflow)**
- ❖ **Операторы независимы**

# Способы построения DAGов

## ❖ Классический

- ✓ создаем операторы
- ✓ используем явную композицию

## ❖ TaskFlow API (2.x)

- ✓ облегченный способ
- ✓ автоматическая композиция
- ✓ только для PythonOperator

# Создание DAG (pipeline)



# Объект: DAG

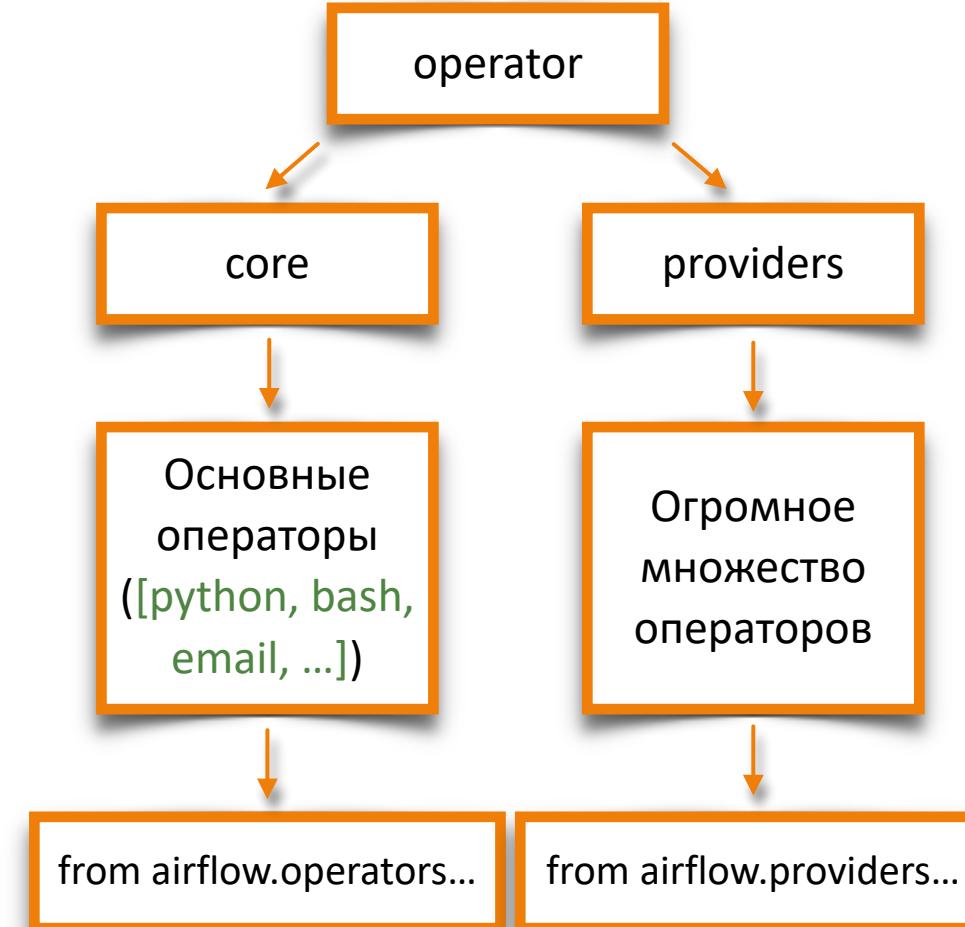
## ❖ Основные атрибуты

- ✓ `name, description`
- ✓ `schedule_interval`
- ✓ `start_date`
- ✓ `trigger_rule`
- ✓ `catchup`
- ✓ `default_args`
- ✓ `tags`
- ✓ `max_active_runs`
- ✓ ...

# Объект: operator

## ❖ Основные атрибуты

- ✓ `id, description`
- ✓ `dag`
- ✓ `start_date`
- ✓ `action`
  - `PythonOperator (some_func)`
  - `BashOperator (script or file)`
  - `HiveOperator (hql)`
  - `[Postgres, MySQL]Operator (sql)`
  - ...
- ✓ ...



# Объект: PythonOperator

## ❖ «Главный» оператор Apache Airflow

### ✓ `callable`

- Определяет обработку данных
- `return None` (как правило)

### ✓ `op_args: positional params` [`«some_param1»`, `some_param2`]

### ✓ `op_kwargs: dict params` {`key: «value»`, `key: «value»`}

# Композиция

## ❖ bitshift композиция (>><<)

```
operator1 >> operator2 >> operator3
```

## ❖ Небольшое усложнение bitshift

```
op1 >> [op2, op3] >> op4
```

## ❖ .set\_[upstream, downstream]()

```
operator1.set_downstream(operator2)  
operator2.set_downstream(operator3)
```

Ну Пожалуйста не надо



ШКОЛА БОЛЬШИХ ДАННЫХ