# DATA ENGINEERING

## Оркестраторы
## Oozie и AirFlow

АРТЁМ БАДАНОВ

Data Engineer

# ОРКЕСТРАТОРЫ

Несмотря на быстрое развитие инструментов потоковой (**streaming**) аналитики, значительная часть расчётов по-прежнему выполняется в пакетном (**batch**) режиме. Это приводит к появлению большого числа повторяющихся задач, которые нужно запускать каждый час/день/месяц. Поэтому у каждой компании, занимающейся обработкой данных, в арсенале есть инструмент, управляющий периодическими задачами. Самый подходящий термин для данных инструментов — **оркестраторы**.

В качестве примеров можно привести несколько сервисов, которые вы уже могли встречать в составе систем обработки данных:

- **Apache Airflow** — пожалуй, наиболее популярная система оркестрации процессов обработки данных на текущий момент. Плюсами являются гибкость, удобство использования и активное развитие.


- **Apache Oozie** — оркестратор, известный благодаря тесной интеграции с **Hadoop-стеком**. Входит в крупнейшие дистрибутивы Hadoop от Cloudera и Hortonworks.

- **Luigi** — ещё один оркестратор, использующий (как и Airflow) Python для описания графов задач.

# Какие задачи решает оркестратор?

Часто оркестратор называют **«распределённым cron'ом»** в честь планировщика **cron** системы Linux.
Это не совсем корректно, поскольку оркестратор выполняет гораздо больше функций:

**Планирование задач**

Основная функция, позволяющая избавиться от ручного запуска рутинных задач по расчёту витрин, загрузке данных, резервному копированию и т. д.

**Управление зависимостями**

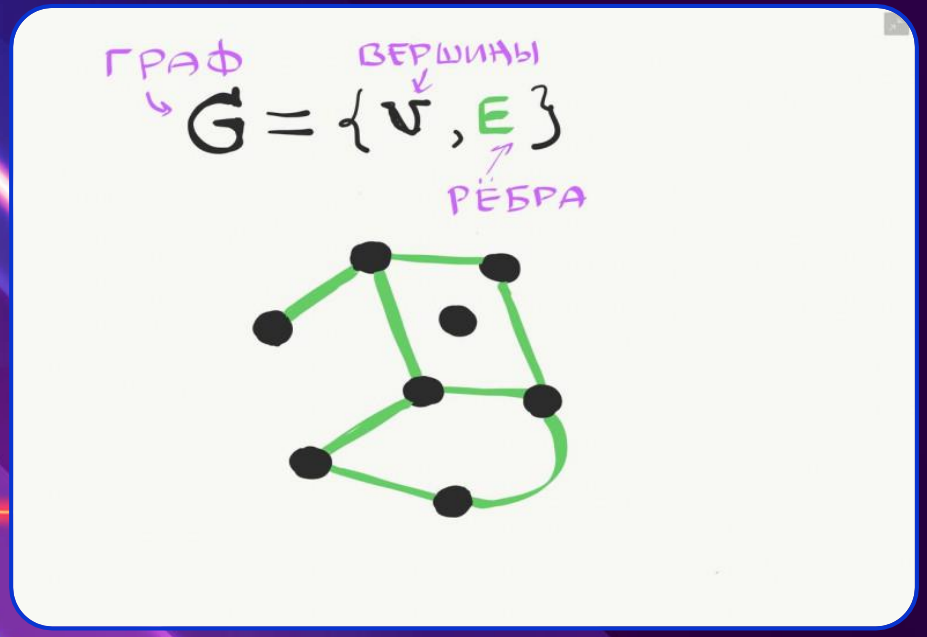
Часто задачу нужно запустить не только в определённый промежуток времени, но и с учётом статуса других задач. Например, расчёт витрины данных нужно запустить только после загрузки сырых данных на кластер.

**Репроцессинг**

Если известно, что какая-то задача требует перезапуска (например, были загружены неполные данные на предыдущем этапе), то перезапуска требуют и задачи, зависящие от неё. Кроме того, перезапуск может быть необходим за несколько временных периодов. В итоге нужно будет руками запустить несколько десятков задач, да ещё и в правильном порядке. Оркестратор позволяет выполнить эту утомительную работу за пару кликов.

# ГРАФЫ

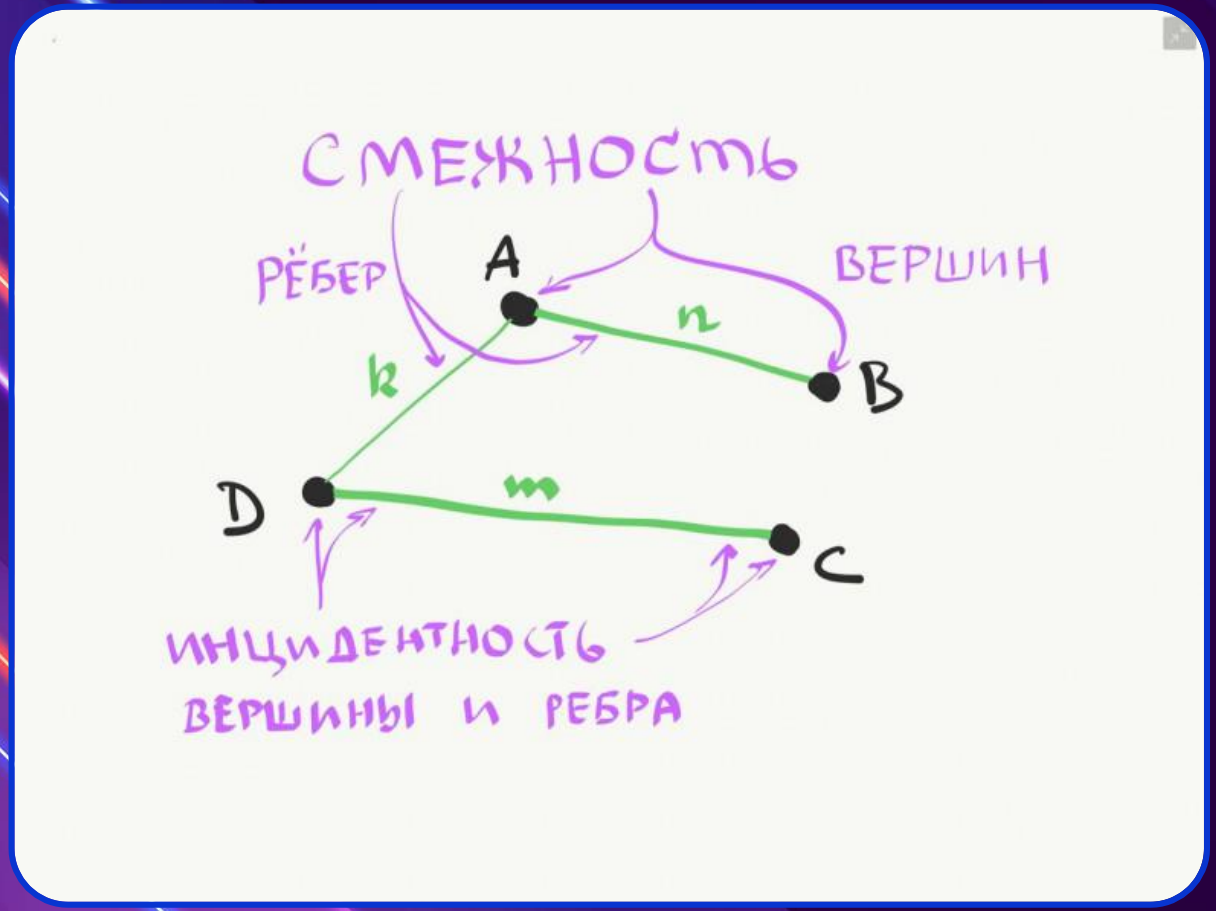
**Граф -** это топологичекая модель, которая состоит из множества вершин и множества соединяющих их рёбер. При этом значение имеет только сам факт, какая вершина с какой соединена.



# ГРАФЫ

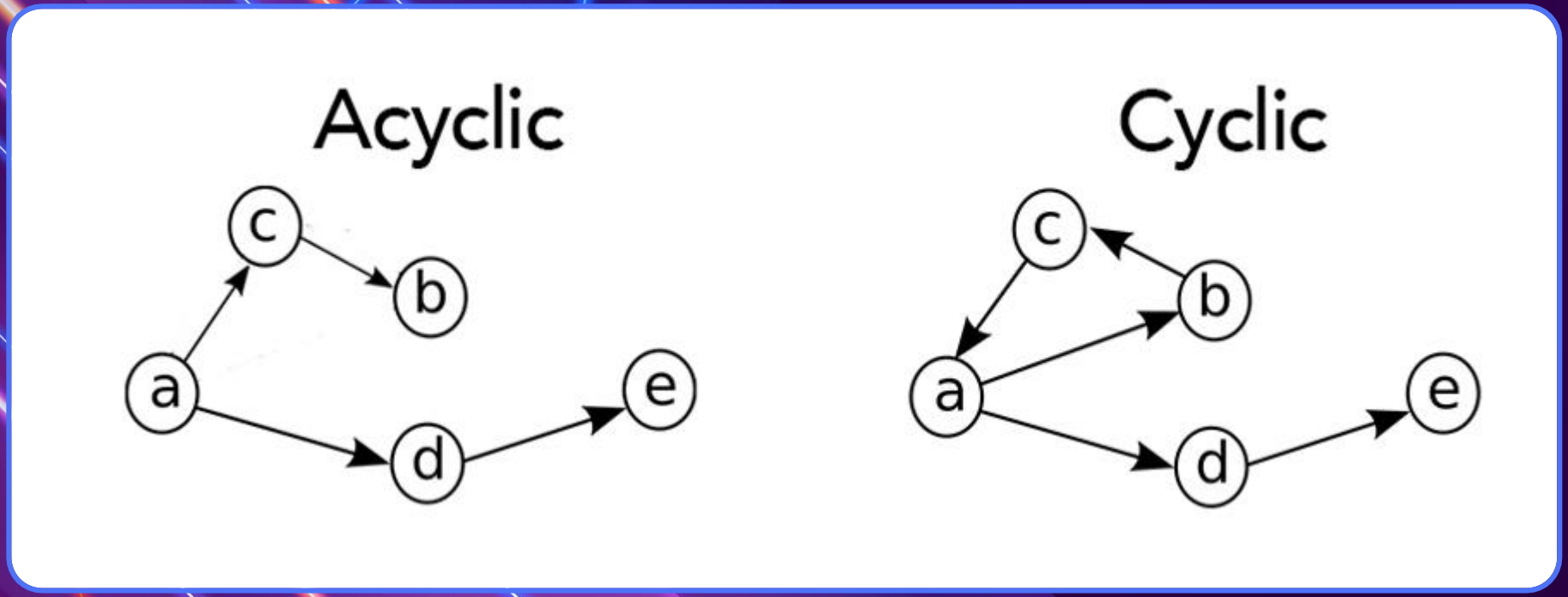
- **Вершина** - точка в графе, отдельный объект, для топологической модели графа не имеет значения координата вершины, её расположение, цвет, вкус, размер; однако при решении некоторых задачах вершины могут раскрашиваться в разные цвета или сохранять числовые значения.
- **Ребро** - неупорядоченная пара двух вершин, которые связаны друг с другом. Эти вершины называются концевыми точками или концами ребра. При этом важен сам факт наличия связи, каким именно образом осуществляется эта связь и по какой дороге - не имеет значения; однако рёбра может быть присвоен «вес», что позволит говорить о «нагруженном графе» и решать задачи оптимизации.
- **Инцидентность** - вершина и ребро называются инцидентными, если вершина является для этого ребра концевой. Обратите внимание, что термин “инцидентность” применим только к вершине и ребру.
- **Смежность вершин** - две вершины называются смежными, если они инцидентны одному ребру.
- **Смежность рёбер** - два ребра называются смежными, если они инцидентно одной вершин

# ГРАФЫ


СМЕЖНОСТЬ
РЁБЕР
ВЕРШИН
A
B
C
D
k
n
m
ИНЦИДЕНТНОСТЬ
ВЕРШИНЫ И РЕБРА

# ОРИЕНТИРОВАННЫЙ АЦИКЛИЧЕСКИЙ ГРАФ

**Ациклическим графом** называется ориентированный граф, не имеющий циклов.



# OOZIE

**Apache Oozie** – это серверная система планирования рабочих процессов для управления заданиями Hadoop. Рабочие процессы в Oozie определяются как набор потоков управления и узлов действий в ориентированном ациклическом графе.

Узлы потока управления определяют начало и конец рабочего процесса(узлы начала, конца и сбоя), а также механизм для контроля пути выполнения рабочего процесса.

# WORKFLOW

Для запуска задачи всегда описывается конфигурация запуска в файле **workflow.xml**

```
<workflow-app name="project-md2-daily-statistic" xmlns="uri:oozie:workflow:0.5">

   <global>
      <configuration>
         <property>
            <name>oozie.launcher.mapred.job.queue.name</name>
            <value>${queue}</value>
         </property>
      </configuration>
   </global>
```

```
   <start to="project-md2-daily-statistic" />

   <action name="project-md2-daily-statistic">
      <spark xmlns="uri:oozie:spark-action:0.1">
         <job-tracker>${jobTracker}</job-tracker>
         <name-node>${nameNode}</name-node>
         <master>yarn-client</master>
         <name>project-md2-daily-statistic</name>
         <class>ru.daily.Statistic</class>
         <jar>${nameNode}${jobDir}/lib/daily-statistic-0.1.jar</jar>
         <spark-opts>
            --queue ${queue}
            --master yarn-client
            --num-executors 5
            --conf spark.executor.cores=8
            --conf spark.executor.memory=10g
            --conf spark.executor.extraJavaOptions=-XX:+UseG1GC
            --conf spark.yarn.jars=*.jar
            --conf spark.yarn.queue=${queue}
         </spark-opts>
         <arg>${nameNode}${dataDir}</arg>
         <arg>${datePartition}</arg>
         <arg>${nameNode}${saveDir}</arg>
      </spark>

      <ok to="end" />
      <error to="fail" />

   </action>

   <kill name="fail">
      <message>Statistics job failed [${wf:errorMessage(wf:lastErrorNode())}]</message>
   </kill>

   <end name="end" />

</workflow-app>
```

# COORDINATOR

Координатор нужен для организации регулярного запуска.
С его помощью можно задавать время и дату запуска.

```
frequency="${frequency}" start="${startTime}" end="${endTime}"
```

Для того чтобы **запускать нашу джобу**, необходимо либо настроить CI/CD, либо воспользоваться командами консоли кластера.

- oozie job -oozie http://hadoop.host.ru:11000/oozie -config coord.properties –run
- oozie job -info {job_id}
- oozie job -kill {job_id}
- oozie jobs -jobtype coordinator -filter user={user_name}

# AIRFLOW

**Apache Airflow** — это решение с открытым исходным кодом. Это оркестратор, который позволяет наладить разработку, планирование и мониторинг сложных рабочих процессов. Также он работает как планировщик ETL/ELT-процессов и использует язык программирования Python.


**В основе концепции Airflow лежит DAG —** направленный ациклический граф.

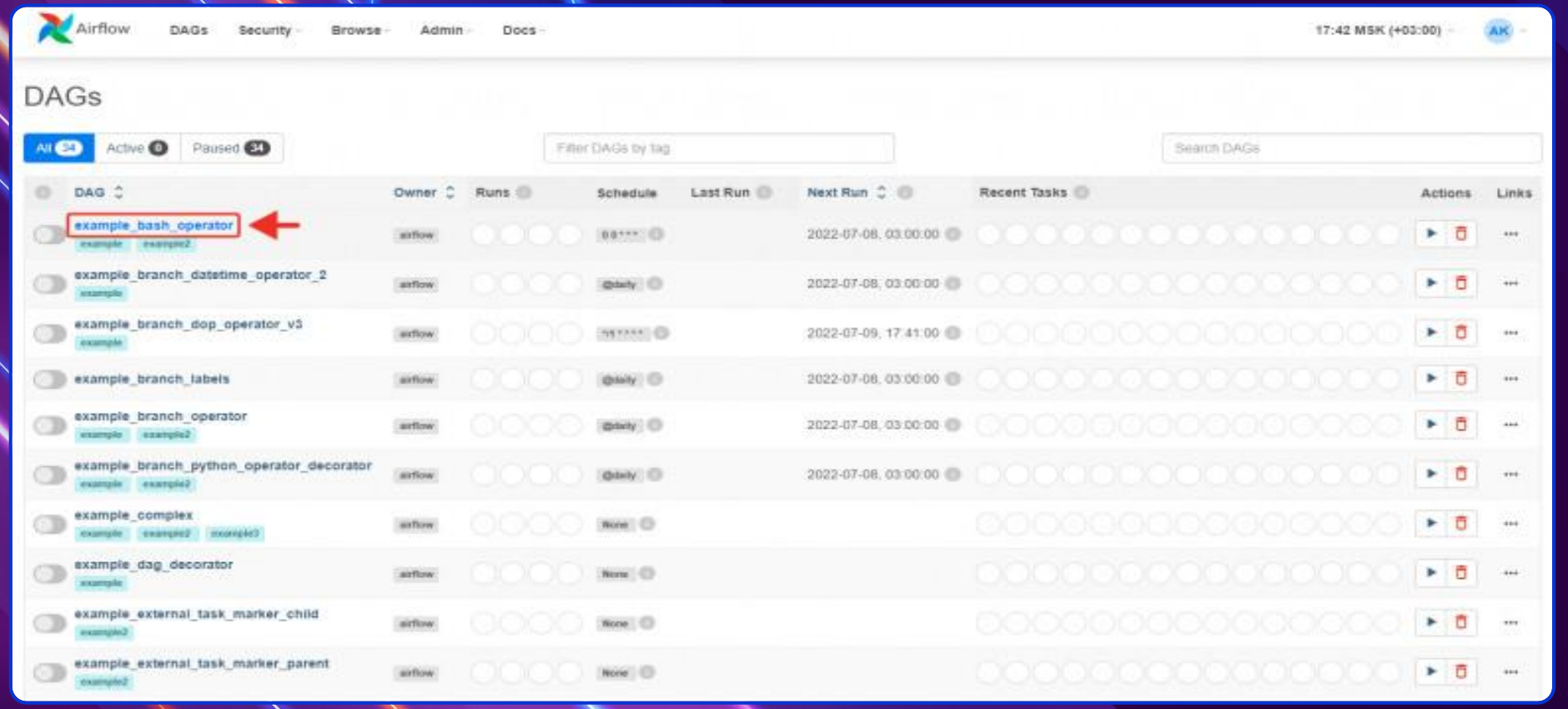
Он описывает процессы обработки данных и позволяет объединять задачи, определяя правила их совместной работы.

# AIRFLOW

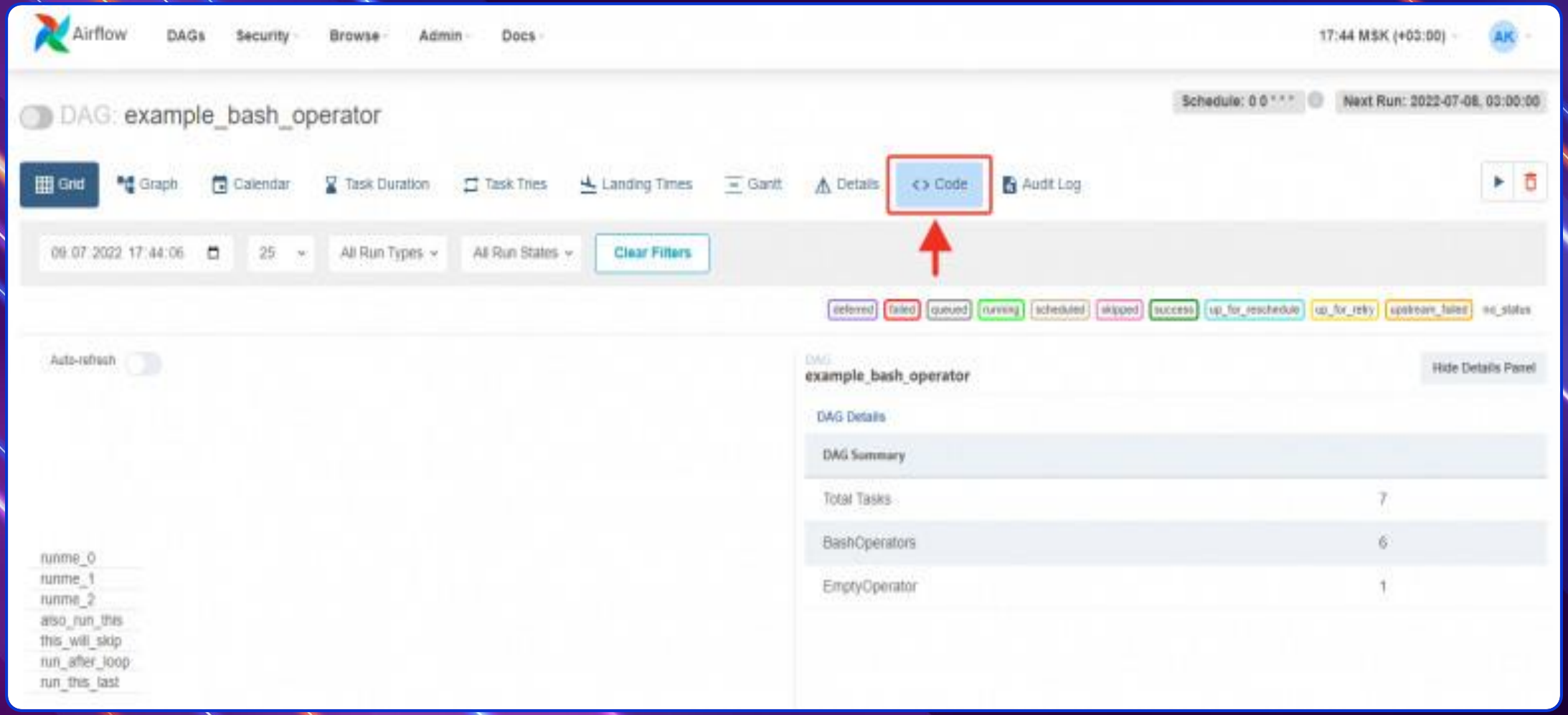
- **Airflow** состоит из нескольких компонентов, но главные из них — scheduler и webserver. Без них ничего не запустится.
- **Scheduler** (планировщик) отслеживает все задачи и DAGs, а затем запускает экземпляры задач после установки их зависимостей. В фоновом режиме планировщик запускает подпроцесс, который отслеживает и синхронизирует все DAGs в указанном каталоге. По умолчанию он раз в минуту собирает результаты синтаксического анализа DAGs и проверяет, необходимо ли запустить какие-либо активные задачи.
- **WebServer** (веб-сервер) отвечает за отображение веб-интерфейса и аутентификацию пользователей, а также решает, к какой группе должен относиться тот или иной пользователь в соответствии с конфигурационным файлом.

# AIRFLOW

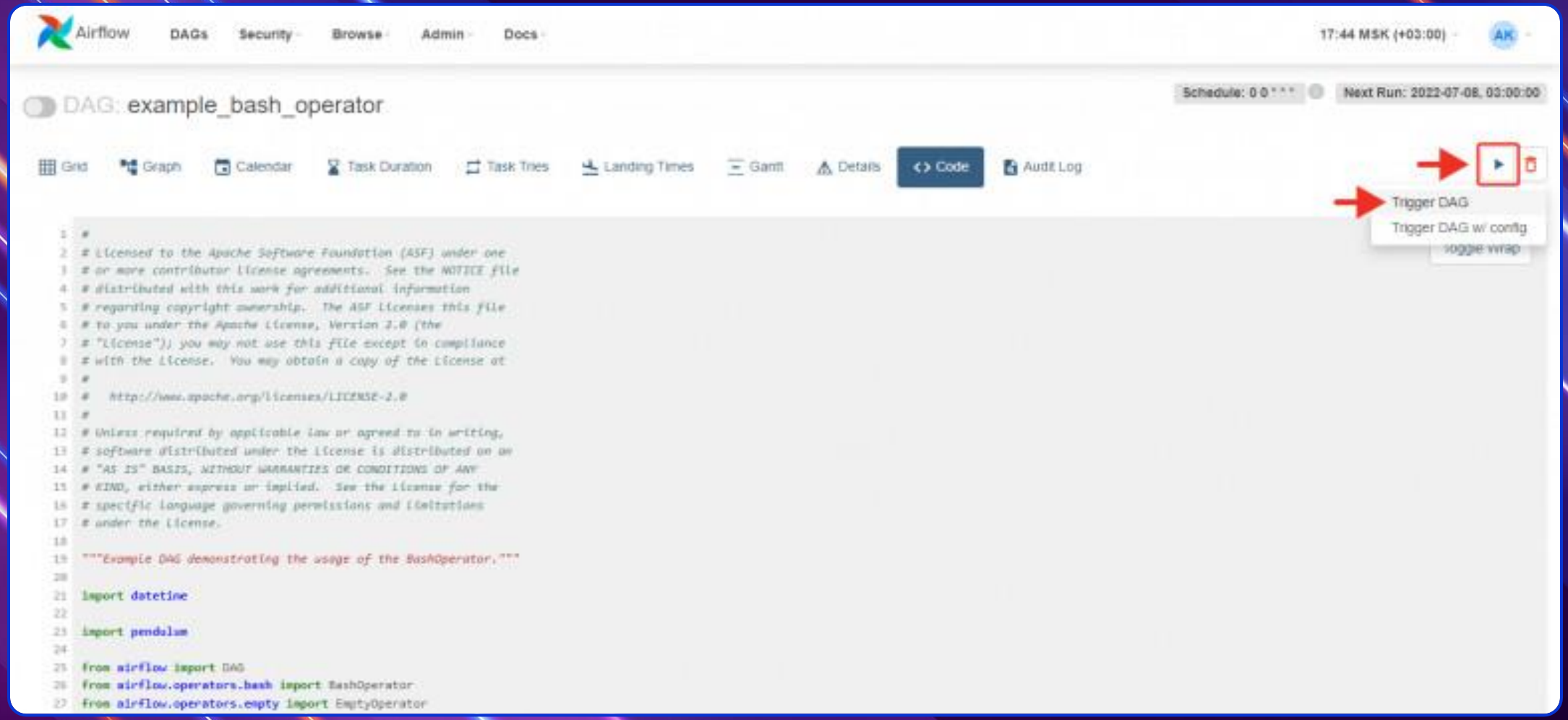
При установке airflow пользователю будет доступно несколько тестовых дагов, предназначенных для демонстрации функциональности. Например, можно зайти в DAG example_bash_operator.



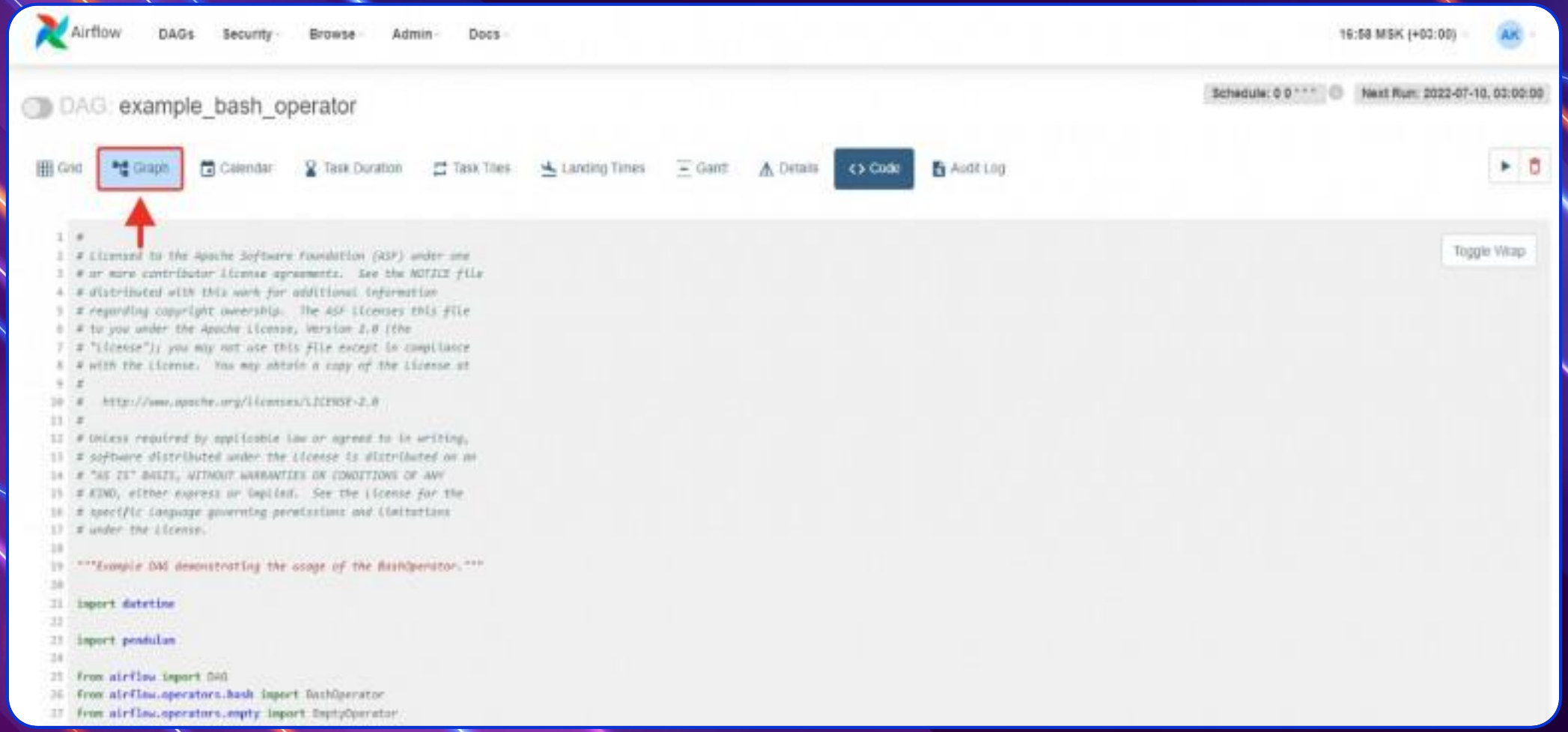
# AIRFLOW


Airflow
DAGs
Security
Browse
Admin
Docs
17:44 MSK (+03:00)
AK
DAG: example_bash_operator
Schedule: 0 0 * * *
Next Run: 2022-07-08, 03:00:00
Grid
Graph
Calendar
Task Duration
Task Tries
Landing Times
Gantt
Details
<> Code
Audit Log
09.07.2022 17:44:06
25
All Run Types
All Run States
Clear Filters
deferred
failed
queued
running
scheduled
skipped
success
up_for_reschedule
up_for_retry
upstream_failed
no_status
Auto-refresh
runme_0
runme_1
runme_2
also_run_this
this_will_skip
run_after_loop
run_this_last
DAG
example_bash_operator
Hide Details Panel
DAG Details
DAG Summary
Total Tasks
7
BashOperators
6
EmptyOperator
1

# AIRFLOW


Airflow
DAGs
Security
Browse
Admin
Docs
17:44 MSK (+03:00)
AK
DAG: example_bash_operator
Schedule: 0 0 * * *
Next Run: 2022-07-08, 03:00:00
Grid
Graph
Calendar
Task Duration
Task Tries
Landing Times
Gantt
Details
Code
Audit Log
Trigger DAG
Trigger DAG w/ config
#
# Licensed to the Apache Software Foundation (ASF) under one
# or more contributor license agreements. See the NOTICE file
# distributed with this work for additional information
# regarding copyright ownership. The ASF licenses this file
# to you under the Apache License, Version 2.0 (the
# "License"); you may not use this file except in compliance
# with the License. You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing,
# software distributed under the License is distributed on an
# "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
# KIND, either express or implied. See the License for the
# specific language governing permissions and limitations
# under the License.
"""Example DAG demonstrating the usage of the BashOperator."""
import datetime
import pendulum
from airflow import DAG
from airflow.operators.bash import BashOperator
from airflow.operators.empty import EmptyOperator

# AIRFLOW


Airflow
DAGs
Security
Browse
Admin
Docs
16:58 MSK (+03:00)
AK
DAG: example_bash_operator
Schedule: 0 0 * * *
Next Run: 2022-07-10, 03:00:00
Grid
Graph
Calendar
Task Duration
Task Tries
Landing Times
Gant
Details
Code
Audit Log
Toggle Wrap
#
# Licensed to the Apache Software Foundation (ASF) under one
# or more contributor license agreements. See the NOTICE file
# distributed with this work for additional information
# regarding copyright ownership. The ASF licenses this file
# to you under the Apache License, Version 2.0 (the
# "License"); you may not use this file except in compliance
# with the License. You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing,
# software distributed under the License is distributed on an
# "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
# KIND, either express or implied. See the License for the
# specific language governing permissions and limitations
# under the License.
"""Example DAG demonstrating the usage of the BashOperator."""
import datetime
import pendulum
from airflow import DAG
from airflow.operators.bash import BashOperator
from airflow.operators.empty import EmptyOperator

# AIRFLOW

**Зеленым** указываются операторы которые должны были отработать и отработали,
а **розовым** то, что не должно.

# СПАСИБО ЗА ВНИМАНИЕ


**Баданов Артем**
**Data Engineer**

**Telegram : @artem5240**
**+7 (977) 699-82-41**