

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.

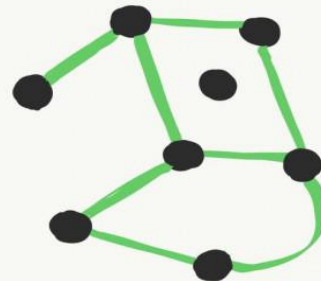




ГРАФЫ

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

ГРАФ
↓
 $G = \{V, E\}$
↑
РЕБРА



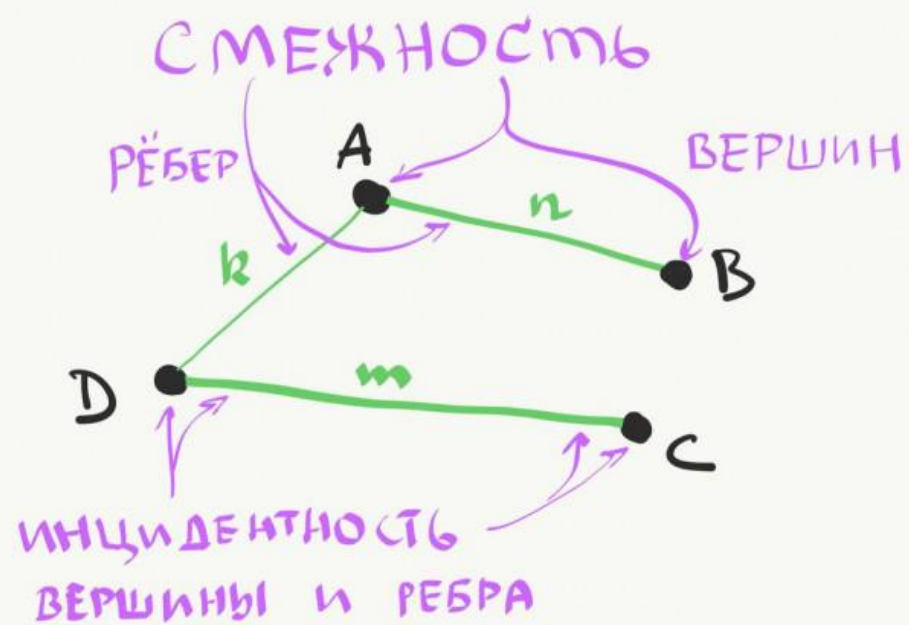


ГРАФЫ

- **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

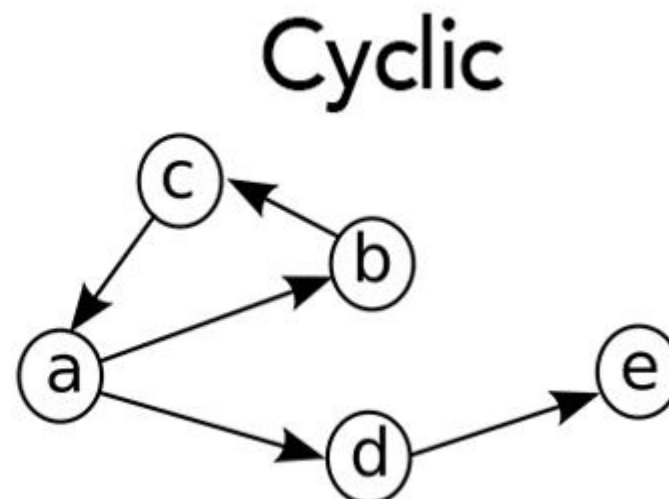
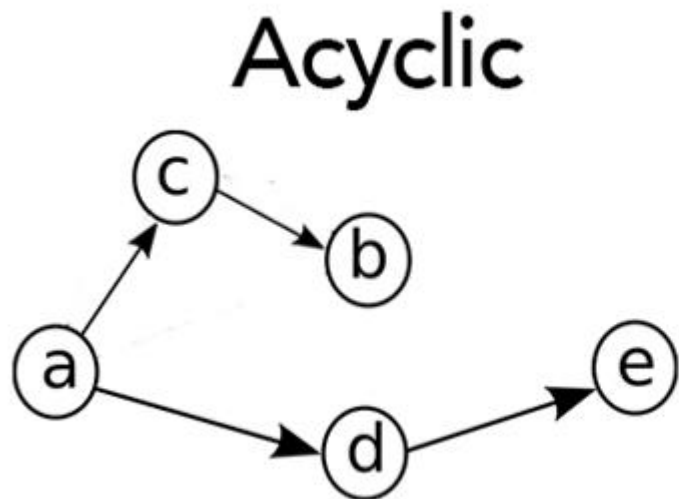


ГРАФЫ



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.



Apache
Airflow

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`.

The screenshot shows the Apache Airflow web interface. At the top, there are navigation links: Airflow, DAGs, Security, Browse, Admin, and Docs. The current page is 'DAGs'. Below the navigation bar, there are filters for 'All' (34), 'Active' (0), and 'Paused' (34). A search bar is also present. The main content is a table of DAGs with columns: DAG, Owner, Runs, Schedule, Last Run, Next Run, Recent Tasks, Actions, and Links. The first row, 'example_bash_operator', is highlighted with a red box and a red arrow pointing to it. Below this row, there are several other DAGs listed, including 'example_branch_datetime_operator_2', 'example_branch_dop_operator_v3', 'example_branch_labels', 'example_branch_operator', 'example_branch_python_operator_decorator', 'example_complex', 'example_dag_decorator', 'example_external_task_marker_child', and 'example_external_task_marker_parent'.

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions	Links
example_bash_operator	airflow	0	@daily		2022-07-08, 03:00:00		[Run] [Refresh] [More]	
example_branch_datetime_operator_2	airflow	0	@daily		2022-07-08, 03:00:00		[Run] [Refresh] [More]	
example_branch_dop_operator_v3	airflow	0	@daily		2022-07-09, 17:41:00		[Run] [Refresh] [More]	
example_branch_labels	airflow	0	@daily		2022-07-08, 03:00:00		[Run] [Refresh] [More]	
example_branch_operator	airflow	0	@daily		2022-07-08, 03:00:00		[Run] [Refresh] [More]	
example_branch_python_operator_decorator	airflow	0	@daily		2022-07-08, 03:00:00		[Run] [Refresh] [More]	
example_complex	airflow	0	None				[Run] [Refresh] [More]	
example_dag_decorator	airflow	0	None				[Run] [Refresh] [More]	
example_external_task_marker_child	airflow	0	None				[Run] [Refresh] [More]	
example_external_task_marker_parent	airflow	0	None				[Run] [Refresh] [More]	



AIRFLOW

The screenshot displays the Apache Airflow web interface. At the top, the navigation bar includes links for DAGs, Security, Browse, Admin, and Docs, along with the current time (17:44 MSK (+03:00)) and a user profile icon (AK). The main header shows the DAG name 'example_bash_operator' and its schedule '0 0 * * *' with the next run time '2022-07-08, 03:00:00'.

Below the header, a row of tabs allows switching between different views: Grid, Graph, Calendar, Task Duration, Task Times, Landing Times, Gantt, Details, **<> Code** (highlighted with a red box and a red arrow), and Audit Log. Below these tabs is a filter bar with a date range '09-07-2022 17:44:06', a page number '25', and dropdowns for 'All Run Types' and 'All Run States', accompanied by a 'Clear Filters' button.

Below the filter bar is a row of colored status tags: colored, failed, queued, running, scheduled, skipped, success, up_for_reschedule, up_for_retry, upstream_failed, and no_status. An 'Auto-refresh' toggle is also present.

The main content area is divided into two panels. The left panel shows a list of task instances: runme_0, runme_1, runme_2, also_run_this, this_will_skip, run_after_loop, and run_this_last. The right panel, titled 'example_bash_operator', contains a 'DAG Summary' table.

DAG Summary	
Total Tasks	7
BashOperators	6
EmptyOperator	1

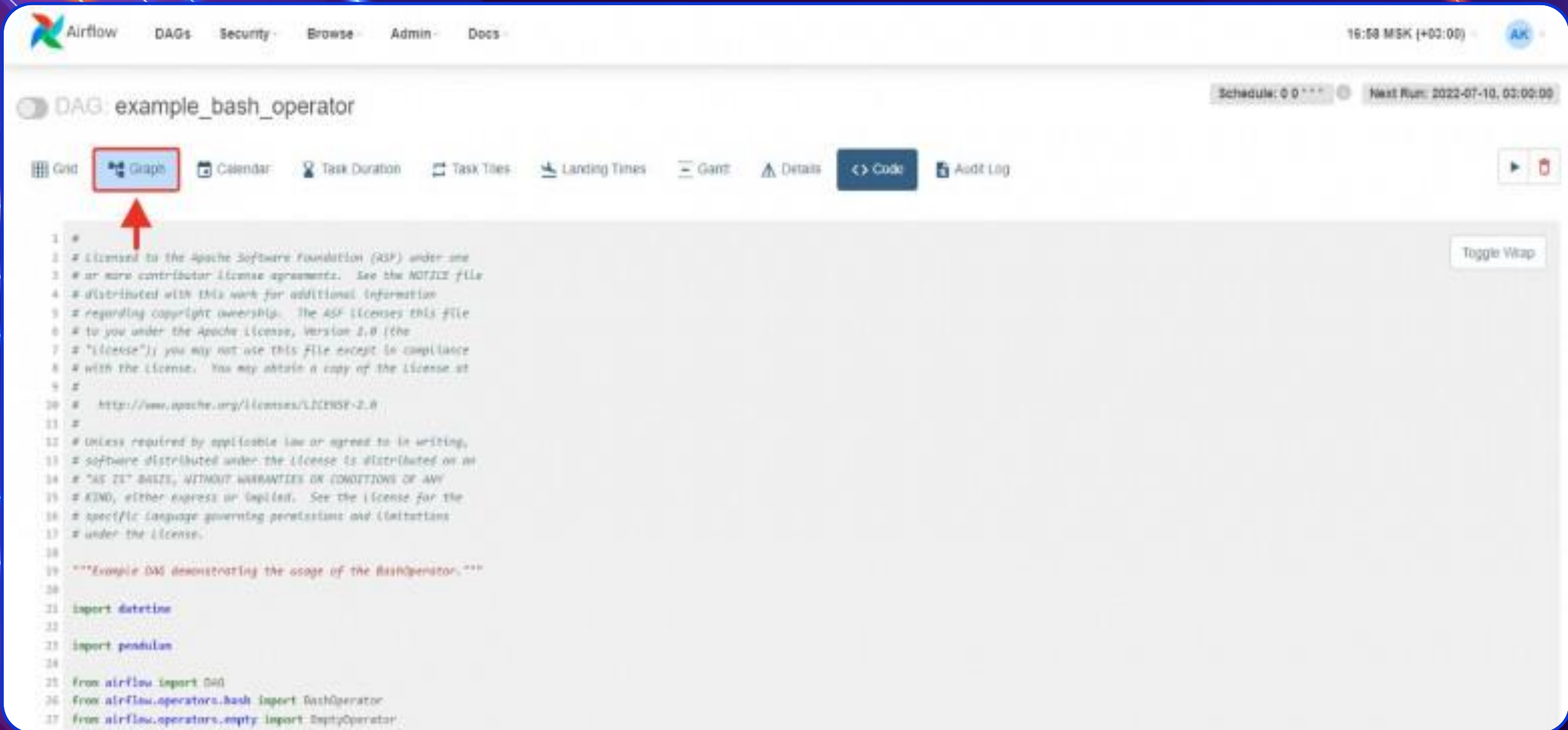
AIRFLOW

The screenshot displays the Apache Airflow web interface. At the top, there's a navigation bar with links for DAGs, Security, Browse, Admin, and Docs. The current time is 17:44 MSK (+03:00) and the user is logged in as AK. The main heading is 'DAG: example_bash_operator'. Below this, there's a toolbar with various views: Grid, Graph, Calendar, Task Duration, Task Times, Landing Times, Gantt, Details, Code (selected), and Audit Log. The 'Code' view shows the DAG's Python code, which includes a license header and imports for datetime, pendulum, and Airflow operators. On the right side, there's a 'Trigger DAG' button with a dropdown menu showing options: 'Trigger DAG', 'Trigger DAG w/ config', and 'Toggle Wrap'. Red arrows point to the 'Trigger DAG' button and its dropdown menu.

Code:

```
1 #
2 # Licensed to the Apache Software Foundation (ASF) under one
3 # or more contributor license agreements. See the NOTICE file
4 # distributed with this work for additional information
5 # regarding copyright ownership. The ASF licenses this file
6 # to you under the Apache license, Version 2.0 (the
7 # "license"); you may not use this file except in compliance
8 # with the license. You may obtain a copy of the license at
9 #
10 # http://www.apache.org/licenses/LICENSE-2.0
11 #
12 # Unless required by applicable law or agreed to in writing,
13 # software distributed under the license is distributed on an
14 # "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
15 # KIND, either express or implied. See the license for the
16 # specific language governing permissions and limitations
17 # under the license.
18
19 """Example DAG demonstrating the usage of the BashOperator."""
20
21 import datetime
22
23 import pendulum
24
25 from airflow import DAG
26 from airflow.operators.bash import BashOperator
27 from airflow.operators.empty import EmptyOperator
```

AIRFLOW



The screenshot displays the Apache Airflow web interface. At the top, the navigation bar includes links for Airflow, DAGs, Security, Browse, Admin, and Docs. The current view is for the DAG 'example_bash_operator'. The top right shows the time '16:58 MSK (+03:00)' and a user profile icon. Below the DAG name, the schedule is '0 0 * * *' and the next run is '2022-07-10, 03:00:00'. The interface has several tabs: Grid, Graph (highlighted with a red box and a red arrow), Calendar, Task Duration, Task Times, Landing Times, Gantt, Details, Code, and Audit Log. The main area shows the DAG code in a text editor. The code includes a license header, imports for 'datetime', 'pendulum', 'DAG', 'BashOperator', and 'EmptyOperator' from 'airflow' and 'airflow.operators.bash', and a comment indicating it's an example DAG.

```
1 #
2 # Licensed to the Apache Software Foundation (ASF) under one
3 # or more contributor license agreements. See the NOTICE file
4 # distributed with this work for additional information
5 # regarding copyright ownership. The ASF licenses this file
6 # to you under the Apache license, version 2.0 (the
7 # "license"); you may not use this file except in compliance
8 # with the license. You may obtain a copy of the license at
9 #
10 # http://www.apache.org/licenses/LICENSE-2.0
11 #
12 # Unless required by applicable law or agreed to in writing,
13 # software distributed under the license is distributed on an
14 # "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
15 # KIND, either express or implied. See the license for the
16 # specific language governing permissions and limitations
17 # under the license.
18
19 """Example DAG demonstrating the usage of the BashOperator."""
20
21 import datetime
22
23 import pendulum
24
25 from airflow import DAG
26 from airflow.operators.bash import BashOperator
27 from airflow.operators.empty import EmptyOperator
```



AIRFLOW

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

Grid Graph Calendar Task Duration Task Times Landing Times Gantt Details <> Code Audit Log

2022-07-09T17:45:59+03:00 Runs 25 Run manual__2022-07-09T14:45:58+00:00 Layout Left > Right Update Find Task...

BashOperator EmptyOperator

deferred failed queued running scheduled skipped success up_for_reschedule up_for_retry upstream_failed no_status

Auto-refresh

```
graph LR; runme_0[runme_0] --> also_run_this[also_run_this]; also_run_this --> run_after_loop[run_after_loop]; runme_1[runme_1] --> run_after_loop; runme_2[runme_2] --> this_will_skip[this_will_skip]; this_will_skip --> run_after_loop; run_after_loop --> run_this_last[run_this_last];
```



СПАСИБО ЗА ВНИМАНИЕ



Баданов Артем
Data Engineer

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

