

[Презентация к уроку 8.4.1](#)

Текстовая расшифровка видео:

КОМПОНЕНТЫ AIRFLOW. WEBVIEW

План:

- **WebView.**

WebView

WebView – это компонент, через который осуществляется управление инстансом Airflow. Чтобы его открыть, необходимо ввести в браузере следующее:

<http://localhost:8080/>

Данные настройки выставлены по умолчанию. Вы можете их изменить в файле конфигурации. О том, как это сделать мы расскажем чуть позже.

Пароль и логин по умолчанию – **Airflow**. Эти настройки задаются в docker-compose-файле.

С запущенным инстансом через docker compose app открываем браузер и видим web-view. Здесь хранятся все DAGs (на данный момент – DAGs-премьеры):

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks
<input type="checkbox"/> dataset_consumes_1 consumes dataset-scheduled	airflow	○○○○○	Dataset		On s3://dag1/output_1.txt	○○○○○
<input type="checkbox"/> dataset_consumes_1_and_2 consumes dataset-scheduled	airflow	○○○○○	Dataset		0 of 2 datasets updated	○○○○○
<input type="checkbox"/> dataset_consumes_1_never_scheduled consumes dataset-scheduled	airflow	○○○○○	Dataset		0 of 2 datasets updated	○○○○○
<input type="checkbox"/> dataset_consumes_unknown_never_scheduled dataset-scheduled	airflow	○○○○○	Dataset		0 of 2 datasets updated	○○○○○
<input type="checkbox"/> dataset_produces_1 dataset-scheduled produces	airflow	○○○○○	@daily		2023-06-25, 00:00:00	○○○○○
<input type="checkbox"/> dataset_produces_2 dataset-scheduled produces	airflow	○○○○○	None			○○○○○
<input type="checkbox"/> example_bash_operator example example2	airflow	○○○○○	00***		2023-06-25, 00:00:00	○○○○○
<input type="checkbox"/> example_branch_datetime_operator example	airflow	○○○○○	@daily		2023-06-25, 00:00:00	○○○○○

Во вкладке DAGs все DAGs, где первая колонка – название **DAG** (это те самые DAG ID, которые мы вводим в Python-файле; далее, **Owner** – владелец (в данном случае – Airflow); **Runs** – это количество запусков, где по «кружкам» отображаются статусы (лучше смотреть один из тех, который уже был запущен):



Здесь количество успешно запущенных. В других кружках – количество запущенных с другими статусами, например, failed:



Здесь также указывается интервал `schedule`, то есть частота повтора DAG. Если `none`, то он не повторяется. Остальные значения означают, с какой периодичностью мы его запускаем:

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks
dataset_consumes_1	airflow	1	Dataset		On s3://dag1/output_1.txt	1
dataset_consumes_1_and_2	airflow	1	Dataset		0 of 2 datasets updated	1
dataset_consumes_1_never_scheduled	airflow	1	Dataset		0 of 2 datasets updated	1
dataset_consumes_unknown_never_scheduled	airflow	1	Dataset		0 of 2 datasets updated	1
dataset_produces_1	airflow	1	@daily		2023-06-25, 00:00:00	1
dataset_produces_2	airflow	1	None			1
example_branch_datetime_operator	airflow	1	0 0 * * *		2023-06-25, 00:00:00	1
example_branch_datetime_operator	airflow	1	@daily		2023-06-25, 00:00:00	1

Last DAG Run – последний запуск информации, то есть информация об этом.

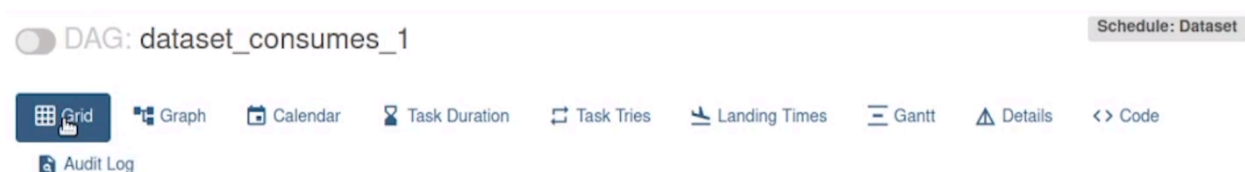
Next Run – планировщик следующего запуска.

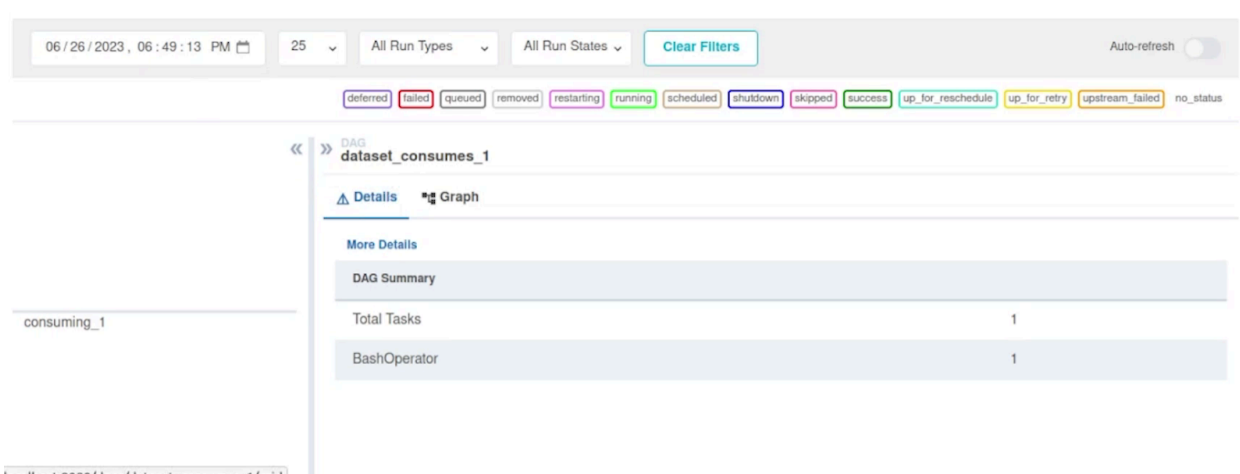
Recent Tasks – это таски, которые исполнялись. Каждый кружочек обозначает статус и количество выполнений.

Это действие, которое можно производить над DAG'ом. Вы можете запустить его вручную отсюда:

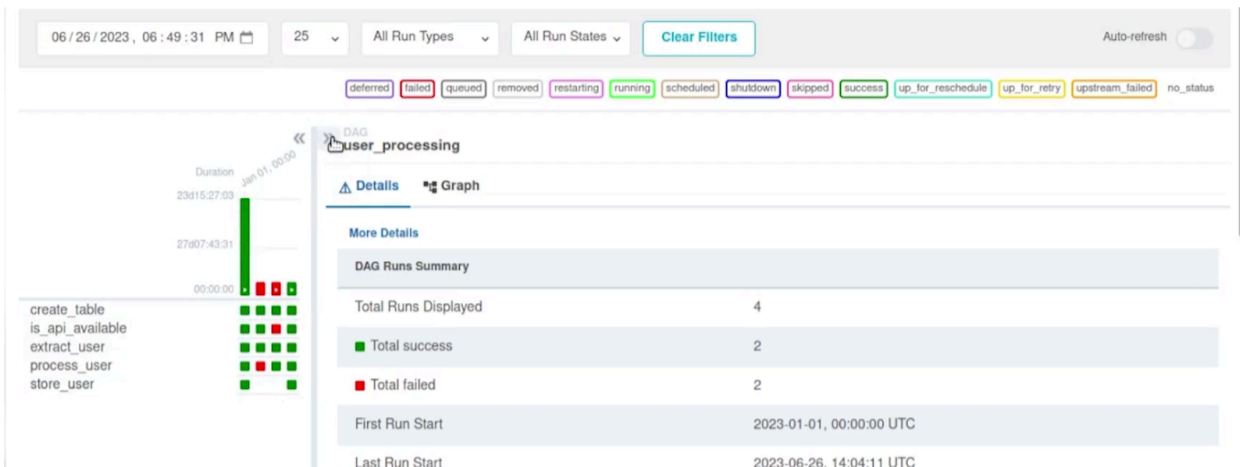
Также вы можете выбрать любое из отображений DAG в этом разделе. Чтобы что-то посмотреть, вам не обязательно заходить в сам DAG.

Рассмотрим один из разделов. Посмотрим, что в форме DAG:

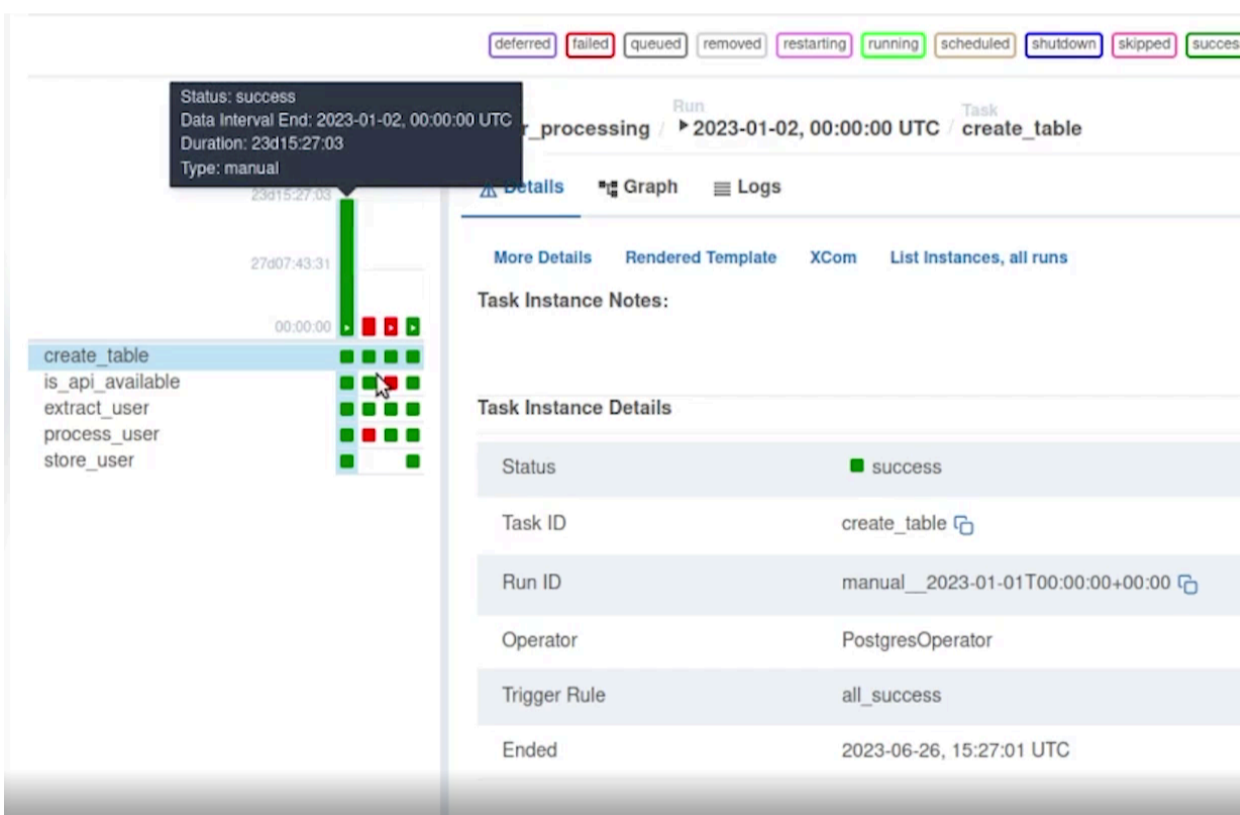




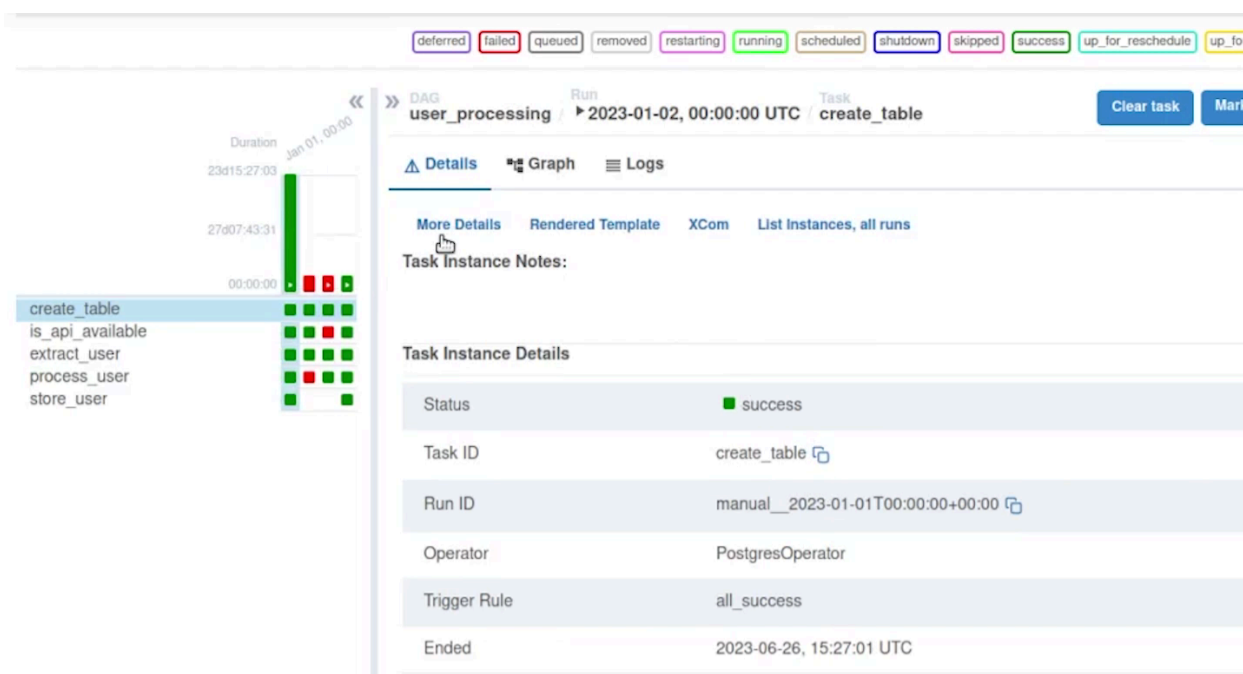
Это **Grid View** – вид-сетка. Посмотрим User Processing. Здесь вид-сетка обозначает следующее:



- Красный цвет – зафейленные задачи, которые были неуспешно выполнены;
- Зеленый цвет – успешно выполненные задачи;
- Вверху располагается полоска для всего DAGs;
- Под серой разделительной полосой находится сетка по задачам. Здесь мы можем смотреть свойства каждого отдельного задачи, точнее Task Instance, запущенного в данное время с такими настройками:



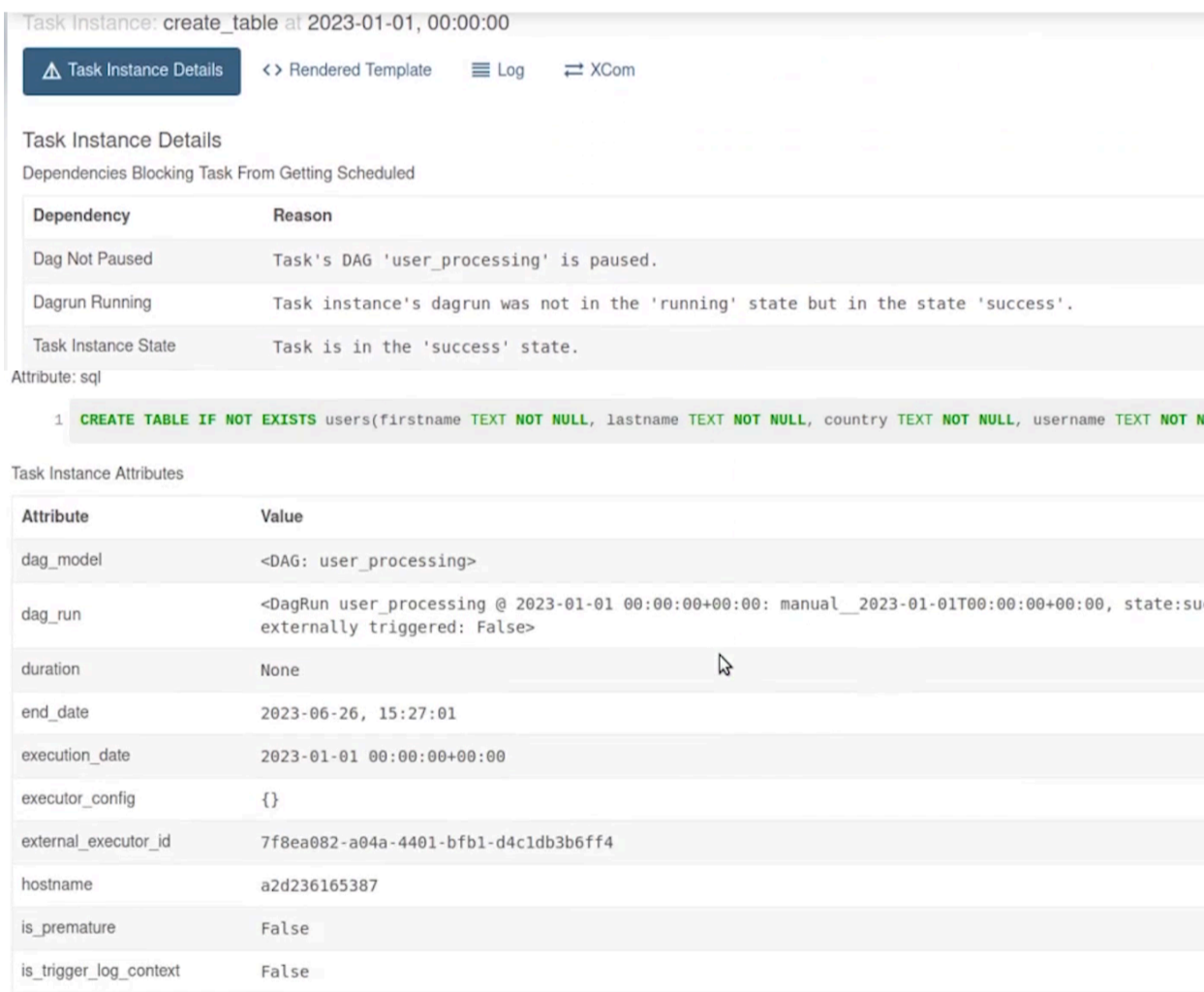
Все это можно спрятать, также можно посмотреть целиком для DAG, то есть для задач и table:



Мы видим следующее:

- Статус задачи;
- Task ID;
- Run ID, где manual означает, что он был запущен вручную;
- Отображение оператора;
- Trigger Rule (почему он запустится; когда он был завершен).

Здесь располагаются все детали по данному задаче:



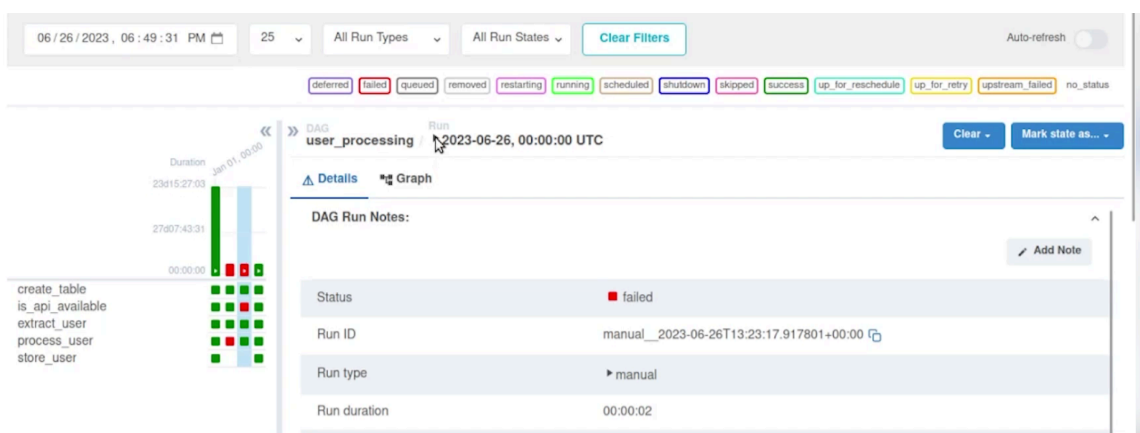
- Из чего он состоит;
- Какие у него есть свойства.

Это статусы задач:

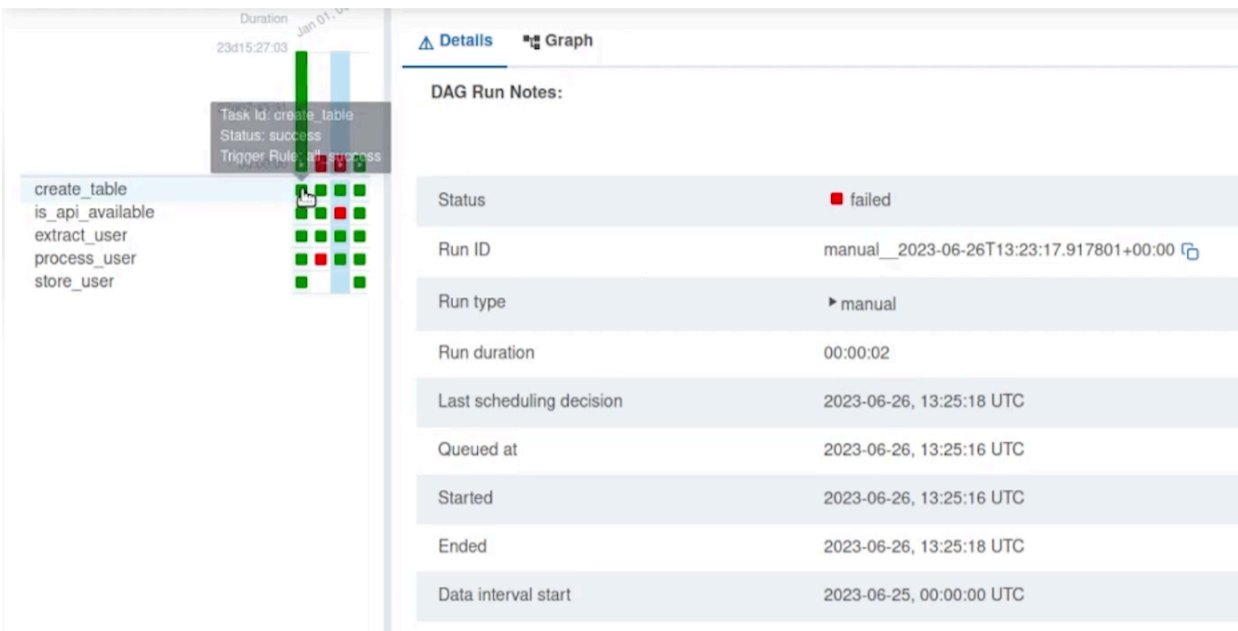


Их можно увидеть, как в Gridview, так в другом формате. В Gridview тоже доступны такие опции.

Если мы нажимаем на «квадратики» слева, то получаем информацию по интересующим нас DAG (если вверху) или по задаче. Здесь есть информация по всему DAG:

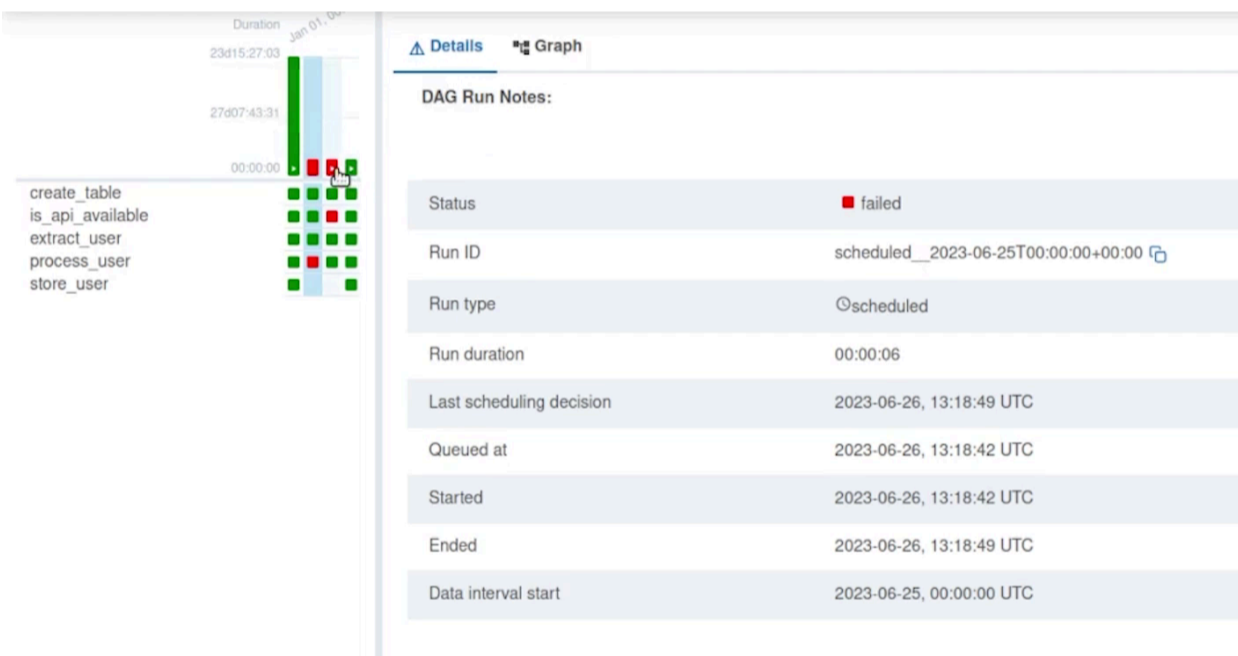


DAG Run-объект – это объект отдельного запуска DAG. По префиксу вначале мы видим «manual», то есть был совершен ручной запуск (или из расписания):



Здесь есть Run ID; Run type указан отдельно со значением «manual».

В данном случае указан scheduled:

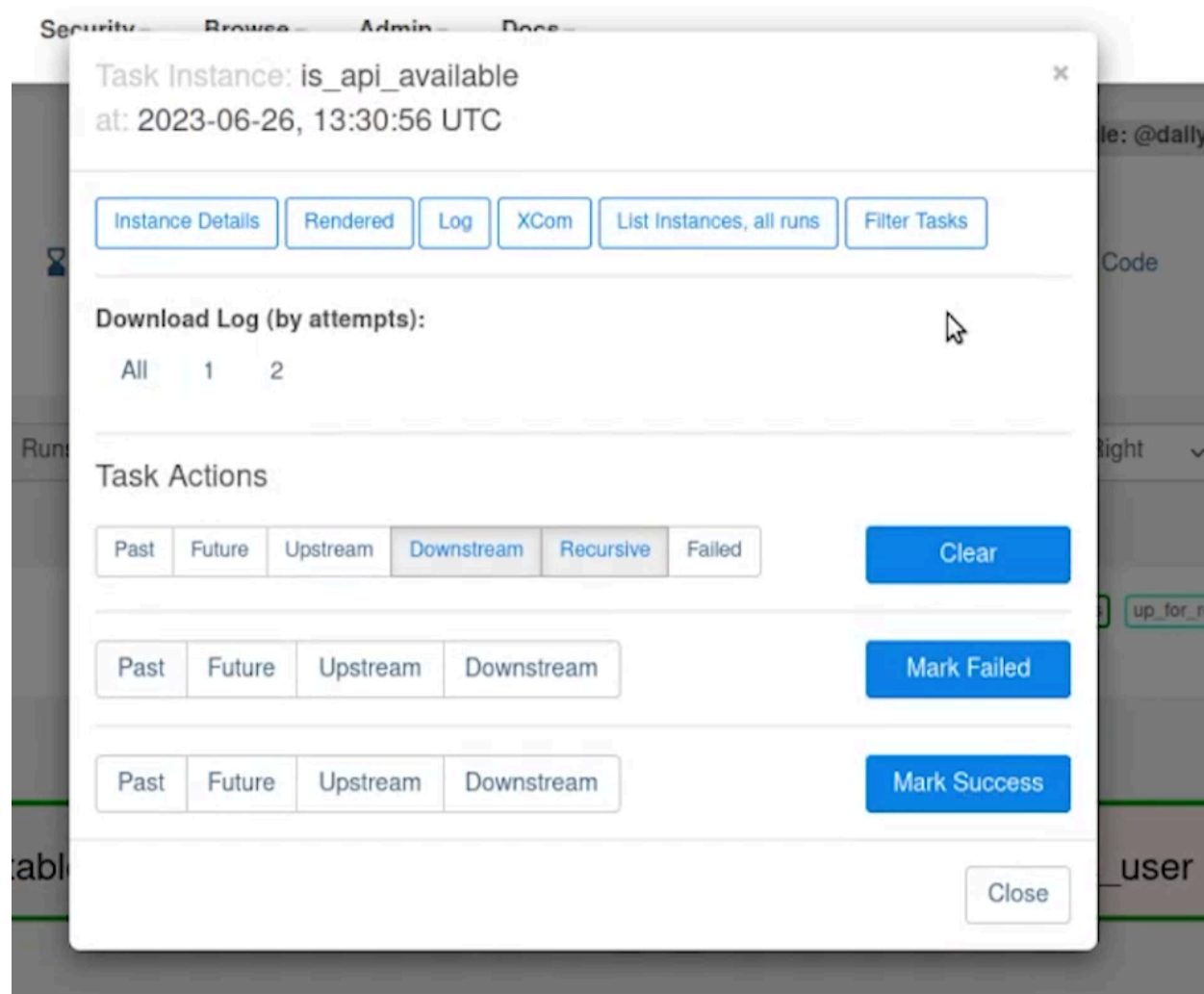


Также мы видим все параметры времени:

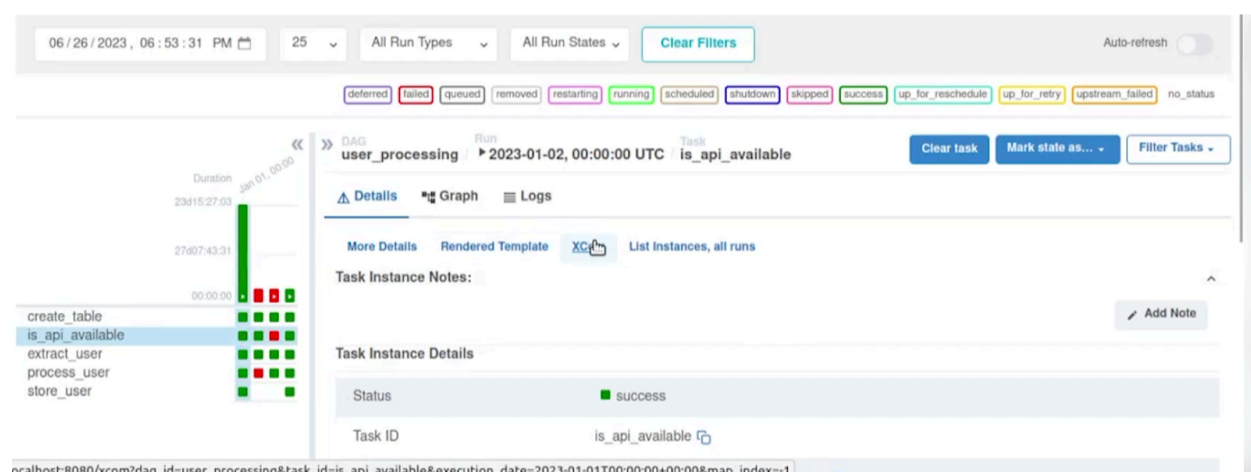
Status	failed
Run ID	manual__2023-06-26T13:23:17.917801+00:00
Run type	manual
Run duration	00:00:02
Last scheduling decision	2023-06-26, 13:25:18 UTC
Queued at	2023-06-26, 13:25:16 UTC
Started	2023-06-26, 13:25:16 UTC
Ended	2023-06-26, 13:25:18 UTC
Data interval start	2023-06-25, 00:00:00 UTC

Перейдем в граф View:

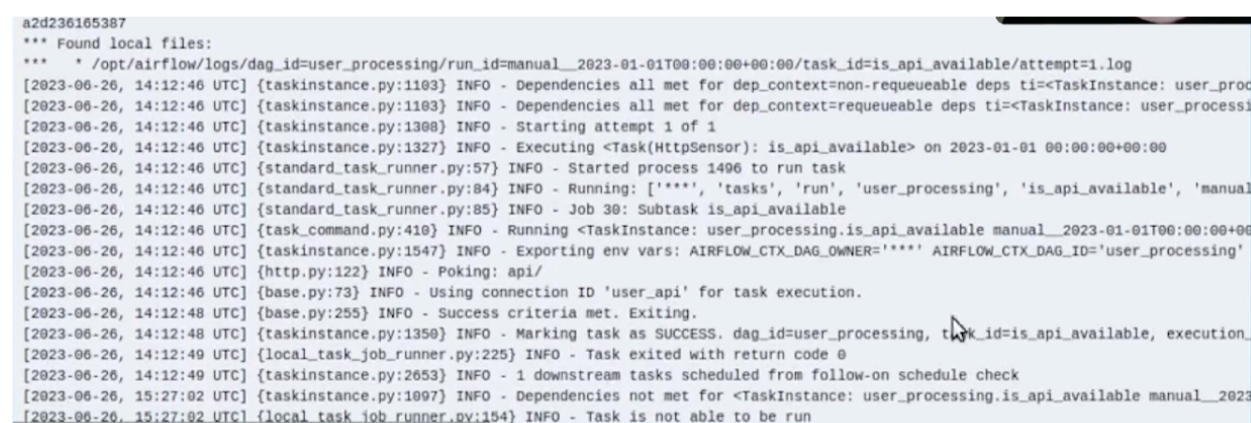
Граф View отображает одновременно один DAG Run. Он очень удобен, чтобы посмотреть информацию по каждому tasku. Здесь можно выбирать какой-то конкретный Run. Удобно и то, что мы можем выбрать конкретный task и посмотреть в нем свойства, или очистить его:



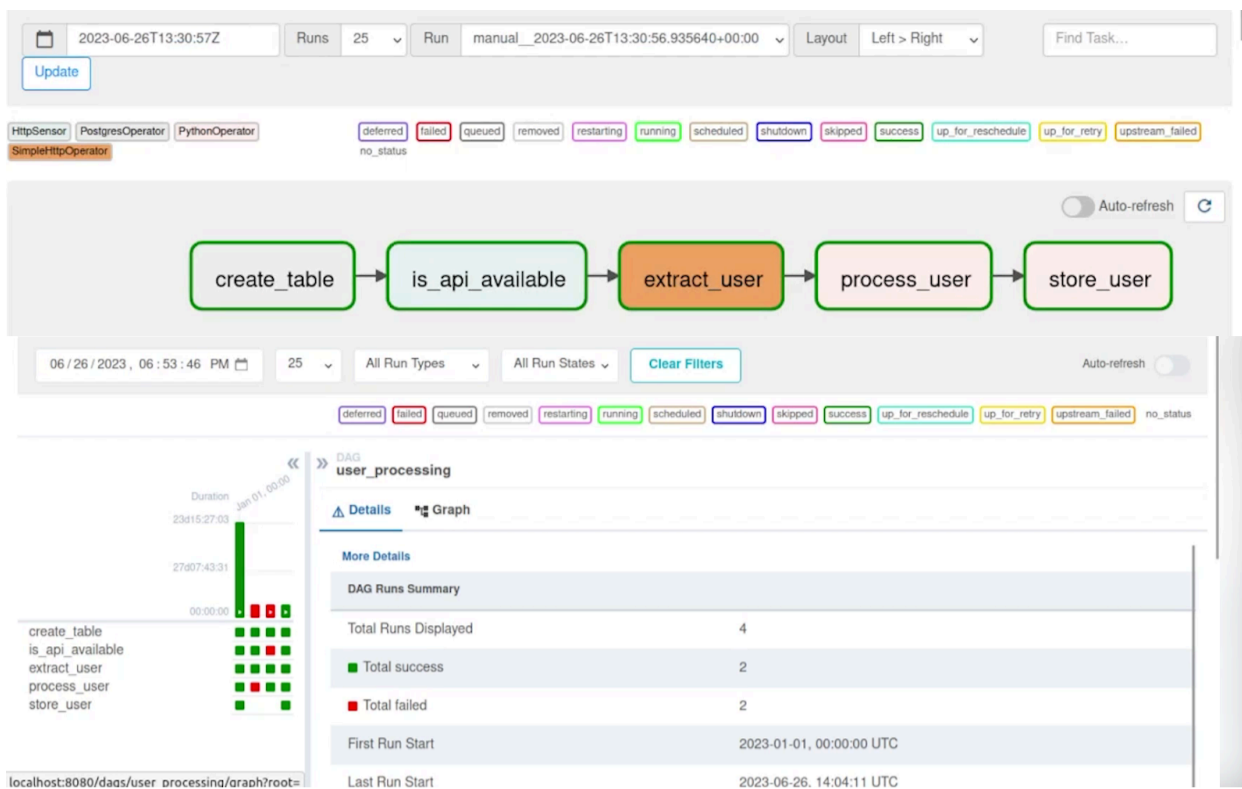
Все эти параметры дублируются. Вы можете смотреть их в **вид-сетка**, если вам так удобнее:



Или в логе:

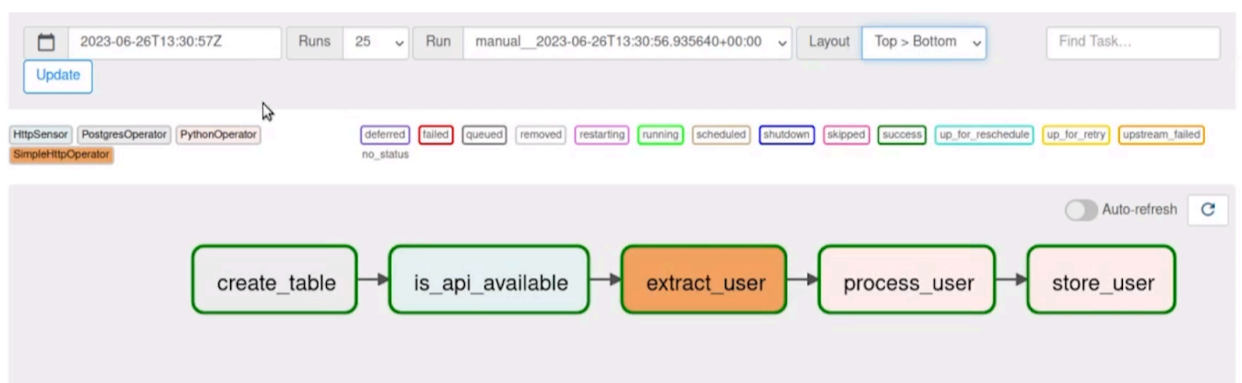


Или в графе:

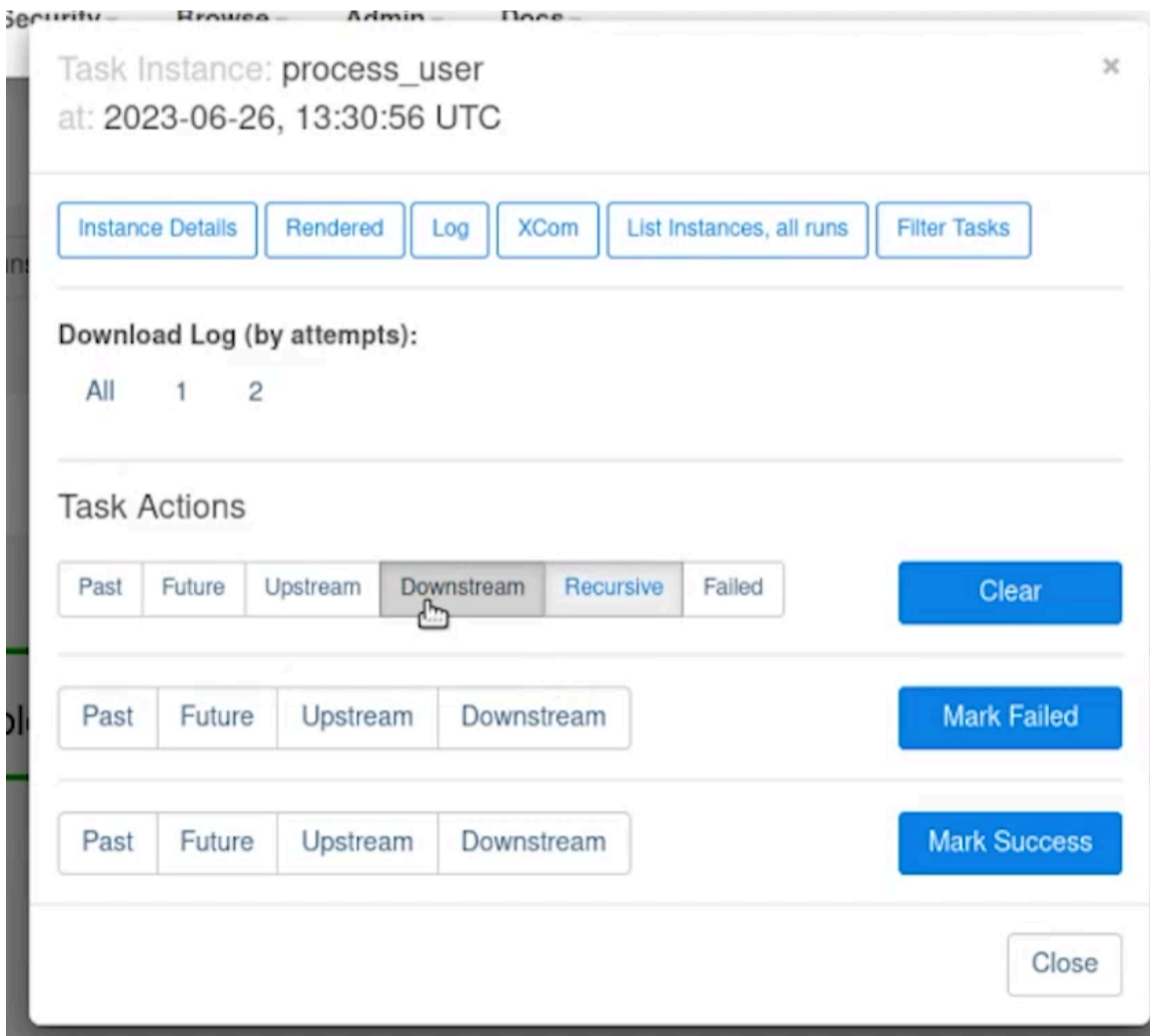


Однако разница все же присутствует:

Из графа удобнее смотреть зависимости, выбирать отдельный DAG Run, выбирать отображения, смотреть статусы. Можно включить автообновление:



Здесь мы можем **очищать**:



Правило: очищаются все последующие. Однако правило можно поменять. Так, очищается вся информация по задаче и сами Task Run'ы. По желанию их можно отметить Failed или Success.

Здесь хранятся все данные о Task instance, как и в Gridview:

Task Instance: process_user at 2023-06-26, 13:30:56

[Task Instance Details](#) [<> Rendered Template](#) [Log](#) [XCom](#)

Task Instance Details

Dependencies Blocking Task From Getting Scheduled

Dependency	Reason
Dag Not Paused	Task's DAG 'user_processing' is paused.
Dagrun Running	Task instance's dagrun was not in the 'running' state but in the state 'success'.
Task Instance State	Task is in the 'success' state.

Task Instance Attributes

Attribute	Value
dag_model	<DAG: user_processing>
dag_run	<DagRun user_processing @ 2023-06-26 13:30:56.935640+00:00: manual_2023-06-26T13:30:56.935640+00:00: queued at: 2023-06-26 14:04:05.019969+00:00. externally triggered: True>

Есть **логи**, также мы можем посмотреть **значение XCom**.

Это еще один способ посмотреть все task-инстансы именно по этому таску:

State	Dag Id	Task Id	Run Id	Map Index	Logical Date	Operator	Start Date
failed	user_processing	process_user	scheduled_2023-06-25T00:00:00+00:00		2023-06-25, 00:00:00	PythonOperator	2023-06-26, 13:18:46
success	user_processing	process_user	manual_2023-06-26T13:23:17.917801+00:00		2023-06-26, 13:23:17	PythonOperator	2023-06-26, 13:25:16
success	user_processing	process_user	manual_2023-06-26T13:30:56.935640+00:00		2023-06-26, 13:30:56	PythonOperator	2023-06-26, 14:04:31
success	user_processing	process_user	manual_2023-01-01T00:00:00+00:00		2023-01-01, 00:00:00	PythonOperator	

Здесь можно оставить либо все предыдущие, либо последующие (как вам удобно):

Task Instance: process_user
at: 2023-06-26, 13:30:56 UTC

[Instance Details](#) [Rendered](#) [Log](#) [XCom](#) [List Instances, all runs](#) [Filter Tasks](#)

Download Log (by attempts):

All	1	2

Task Actions

Past Future Upstream **Downstream** Recursive Failed [Clear](#)

Past Future Upstream Downstream [Mark Failed](#)

Past Future Upstream Downstream [Mark Success](#)

[Close](#)

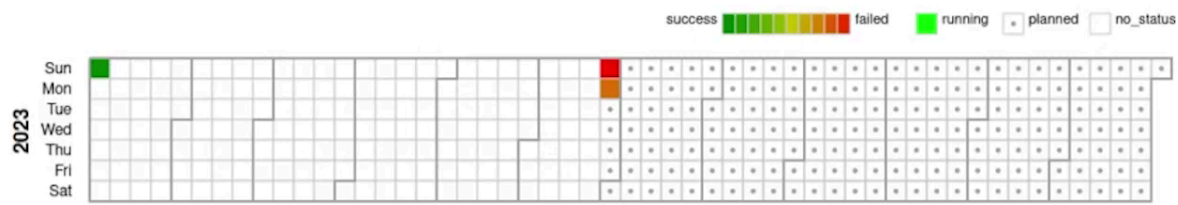
2023-06-26T13:30:57Z Runs 25 Run manual_2023-06-26T13:30:56.935640+00:00 Layout Left > Right Find Task... [Update](#)

HttpSensor PostgresOperator PythonOperator SimpleHttpOperator deferred failed queued removed restarting running scheduled shutdown skipped success up_for_reschedule up_for_retry upstream_failed no_status

Auto-refresh [Refresh](#)

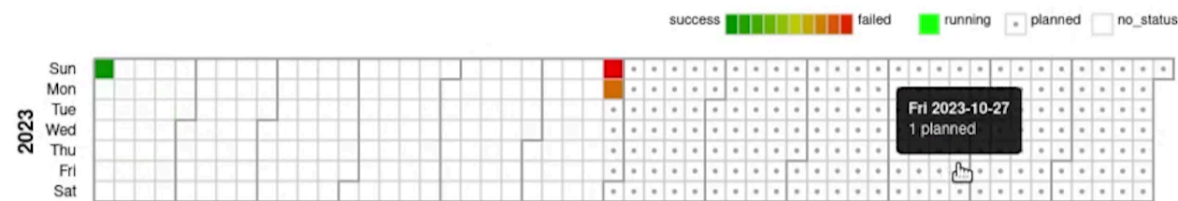
```
graph LR; create_table --> is_api_available; is_api_available --> extract_user; extract_user --> process_user;
```

Вид-календарь.

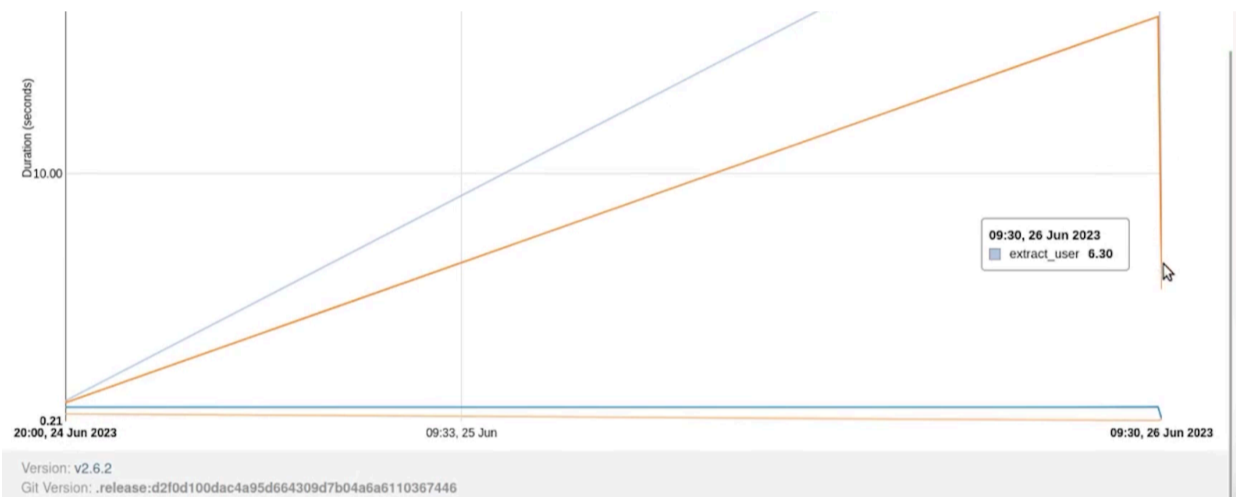


Вид-календарь удобен тем, что можно посмотреть по календарю в какие дни особенно бывает просадка или много падений задач. Например, если это выходные, то можно предусмотреть. Чем квадратик более зеленый, тем больший процент задач выполнялся успешно.

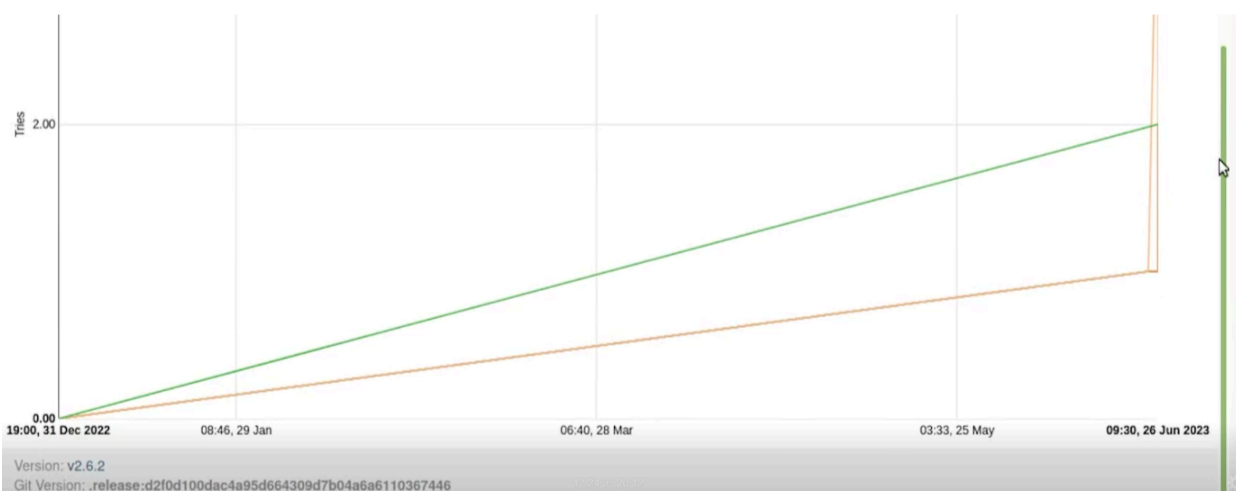
Точки в середине квадратов – это запланированные на будущее задачи:



Task Duration – это время длительности исполнения тасков по датам:



Task Tries – это то, сколько обычно таски делают попыток:



Landing Times замеряет время выполнения тасков с момента запуска DAG, то есть с момента, когда был сдвинут ползунок DAG:

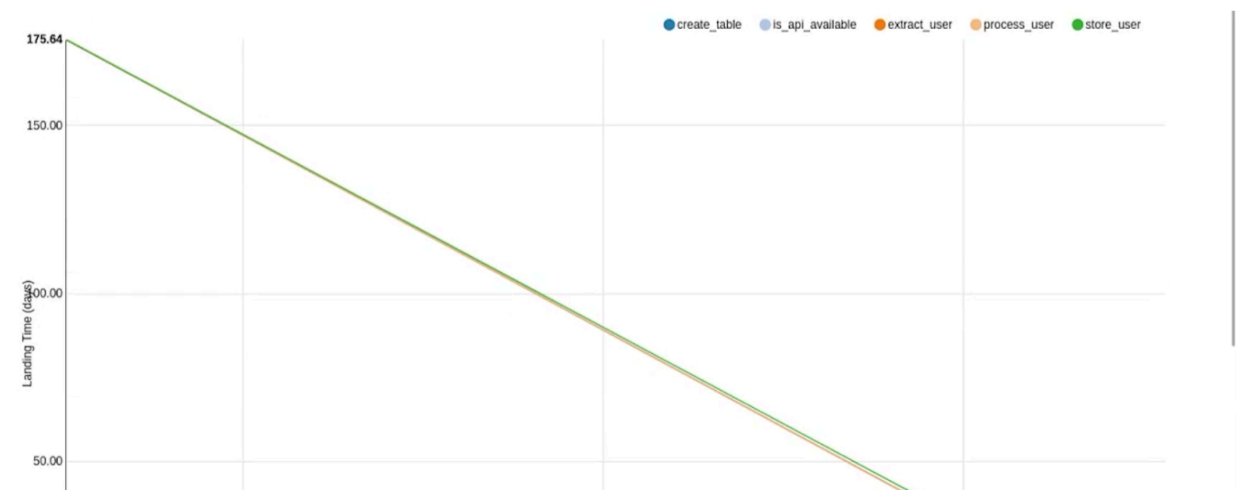
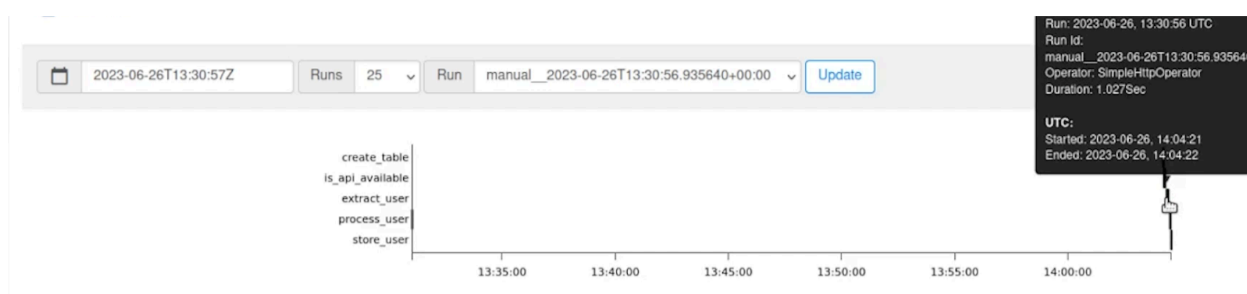


Диаграмма Gantt показывает длительность выполнения каждого таска. Если видим очень длинную зеленую полосу, то это значит, что какой-то из тасков замедляет выполнение DAG. Это так называемое «Бутылочное горлышко», на Landing Times это тоже можно увидеть:



С этим можно что-то сделать. Это менее очевидно, чем статус «Failed», поскольку не всегда при большом количестве задач можно отследить время выполнения.

Здесь можно посмотреть диаграмму Gantt для какого-то конкретного DAG Run.

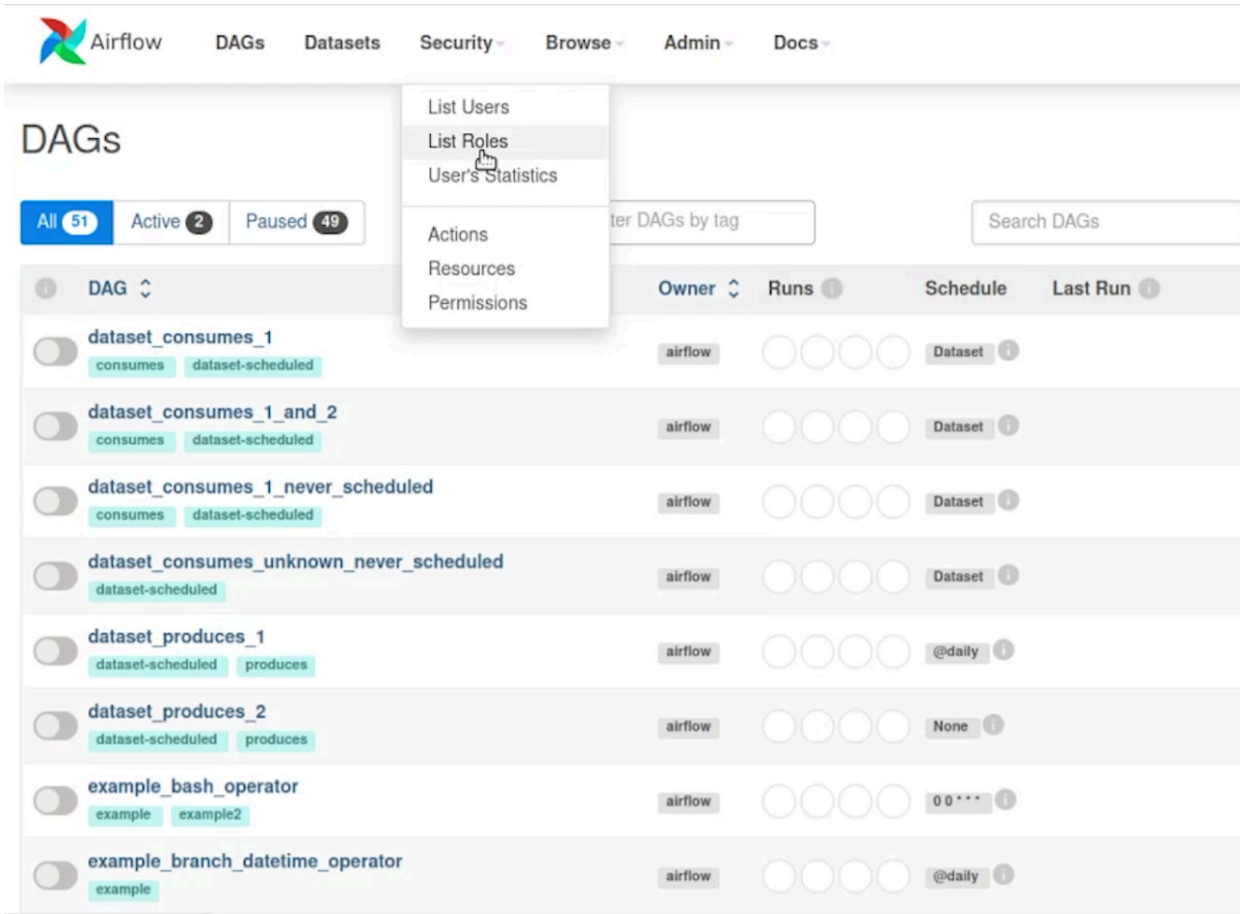
Это **Details**, к которым есть доступ в Gridview в сетке. Так может быть удобнее:

Code. Здесь удобно смотреть внеслись ли изменения, которые вы добавили в файл в директории DAGs и др. Это более быстрый доступ:

```
15 processed_user = json_normalize({'firstname': user['name']['first'],
16                               'lastname': user['name']['last'],
17                               'country': user['location']['country'],
18                               'username': user['login']['username'],
19                               'password': user['login']['password'],
20                               'email': user['email']})
21 processed_user.to_csv('/tmp/processed_user.csv', index=None, header=False)
22
23 def _store_user():
24     hook = PostgresHook(postgres_conn_id='postgres',)
25     hook.copy_expert(sql="COPY users FROM stdin WITH DELIMITER as ',' ", filename='/tmp/processed_user.csv')
26
27
28 with DAG('user_processing', start_date=datetime(2023,1,1), schedule_interval='@daily', catchup=False) as dag:
29
30
31     create_table = PostgresOperator(task_id='create_table', postgres_conn_id='postgres',
32                                   sql='''CREATE TABLE IF NOT EXISTS users(firstname TEXT NOT NULL, lastname TEXT NOT NULL,
33
34
35     is_api_available = HttpSensor(task_id='is_api_available', http_conn_id='user_api', endpoint='api/')
36
37     extract_user = SimpleHttpOperator(task_id='extract_user', http_conn_id='user_api', endpoint='api/', method='GET',
38                                     response_filter=lambda response: json.loads(response.text))
39
40     process_user = PythonOperator(task_id='process_user', python_callable=_process_user, )
41
42     store_user = PythonOperator(task_id='store_user', python_callable=_store_user)
43
44 create_table >> is_api_available >> extract_user >> process_user >> store_user
```

Если мы нажмем на «**корзину**», то у нас очистятся все DAG Run'ы и вся информация, кроме самого текста DAG. Его можно будет запустить спустя какое-то время.

Здесь хранятся юзеры инстанса и их роли. Так, например, если Airflow-инстансом пользуются несколько отделов, их можно разграничить по ролям, чтобы к некоторым DAG не было доступа:



localhost:8080/roles/list/

DAG Runs – это инстансы запуска DAG, то есть конкретные запуски. Их можно посмотреть по полям, также можно удалить, если что-то пошло не так (изменения, которые вносил DAG удаляться не будут):

State	Dag Id	Logical Date	Run Id	Run Type	Queued At	Start Date	End
success	user_processing	2023-06-26, 13:30:56	manual__2023-06-26T13:30:56.935640+00:00	manual	2023-06-26, 14:04:05	2023-06-26, 14:04:11	2023
failed	user_processing	2023-06-26, 13:23:17	manual__2023-06-26T13:23:17.917801+00:00	manual	2023-06-26, 13:25:16	2023-06-26, 13:25:16	2023
success	example_task_group	2023-06-26, 16:09:32	scheduled__2023-06-25T16:09:32.437293+00:00	scheduled	2023-06-26, 16:09:40	2023-06-26, 16:09:40	2023
success	xcom_edu_dag	2023-06-25, 00:00:00	scheduled__2023-06-25T00:00:00+00:00	scheduled	2023-06-26, 16:07:12	2023-06-26, 16:11:23	2023
failed	user_processing	2023-06-25, 00:00:00	scheduled__2023-06-25T00:00:00+00:00	scheduled	2023-06-26, 13:18:42	2023-06-26, 13:18:42	2023
success	user_processing	2023-01-01, 00:00:00	manual__2023-01-01T00:00:00+00:00	manual		2023-01-01, 00:00:00	2023

Можно посмотреть Job'ы. **Job'ы** – это конкретные процессы, которые выполняются на машине:

Id	Dag Id	State	Job Type	Start Date	End Date	Latest Heartbeat	Executor Class	Hostname	Unixname
40	xcom_edu_dag	success	LocalTaskJob	2023-06-26, 16:11:26	2023-06-26, 16:11:27	2023-06-26, 16:11:26	CeleryExecutor	7b8c5176d225	airflow
39	xcom_edu_dag	success	LocalTaskJob	2023-06-26, 16:11:24	2023-06-26, 16:11:25	2023-06-26, 16:11:24	CeleryExecutor	7b8c5176d225	airflow
38	example_task_group	success	LocalTaskJob	2023-06-26, 16:09:44	2023-06-26, 16:09:45	2023-06-26, 16:09:44	CeleryExecutor	7b8c5176d225	airflow
37	example_task_group	success	LocalTaskJob	2023-06-26, 16:09:43	2023-06-26, 16:09:43	2023-06-26, 16:09:43	CeleryExecutor	7b8c5176d225	airflow
36	xcom_edu_dag	success	LocalTaskJob	2023-06-26, 16:04:32	2023-06-26, 16:04:32	2023-06-26, 16:04:32	CeleryExecutor	7b8c5176d225	airflow
35	xcom_edu_dag	success	LocalTaskJob	2023-06-26, 16:04:29	2023-06-26, 16:04:30	2023-06-26, 16:04:29	CeleryExecutor	7b8c5176d225	airflow
34	user_processing	success	LocalTaskJob	2023-06-26, 15:27:04	2023-06-26, 15:27:04	2023-06-26, 15:27:04	CeleryExecutor	7b8c5176d225	airflow
33	user_processing	success	LocalTaskJob	2023-06-26, 15:27:02	2023-06-26, 15:27:02	2023-06-26, 15:27:02	CeleryExecutor	7b8c5176d225	airflow
32	user_processing	success	LocalTaskJob	2023-06-26, 15:27:02	2023-06-26, 15:27:02	2023-06-26, 15:27:02	CeleryExecutor	7b8c5176d225	airflow
31	user_processing	success	LocalTaskJob	2023-06-26, 14:12:50	2023-06-26, 14:12:50	2023-06-26, 14:12:50	CeleryExecutor	7b8c5176d225	airflow
30	user_processing	success	LocalTaskJob	2023-06-26, 14:12:46	2023-06-26, 14:12:49	2023-06-26, 14:12:46	CeleryExecutor	7b8c5176d225	airflow

Task instance – это те самые задачи, которые обрабатывают:

State	Dag Id	Task Id	Run Id	Map Index	Logical Date	Operator
success	user_processing	create_table	scheduled__2023-06-25T00:00:00+00:00		2023-06-25, 00:00:00	PostgresOp
success	user_processing	extract_user	scheduled__2023-06-25T00:00:00+00:00		2023-06-25, 00:00:00	SimpleHttpC
failed	user_processing	process_user	scheduled__2023-06-25T00:00:00+00:00		2023-06-25, 00:00:00	PythonOper
success	user_processing	is_api_available	scheduled__2023-06-25T00:00:00+00:00		2023-06-25, 00:00:00	HttpSensor
success	user_processing	create_table	manual__2023-06-26T13:23:17.917801+00:00		2023-06-26, 13:23:17	PostgresOp
failed	user_processing	is_api_available	manual__2023-06-26T13:23:17.917801+00:00		2023-06-26, 13:23:17	HttpSensor
success	user_processing	extract_user	manual__2023-06-26T13:23:17.917801+00:00		2023-06-26, 13:23:17	SimpleHttpC
success	user_processing	process_user	manual__2023-06-26T13:23:17.917801+00:00		2023-06-26, 13:23:17	PythonOper
success	user_processing	create_table	manual__2023-06-26T13:30:56.935640+00:00		2023-06-26, 13:30:56	PostgresOp

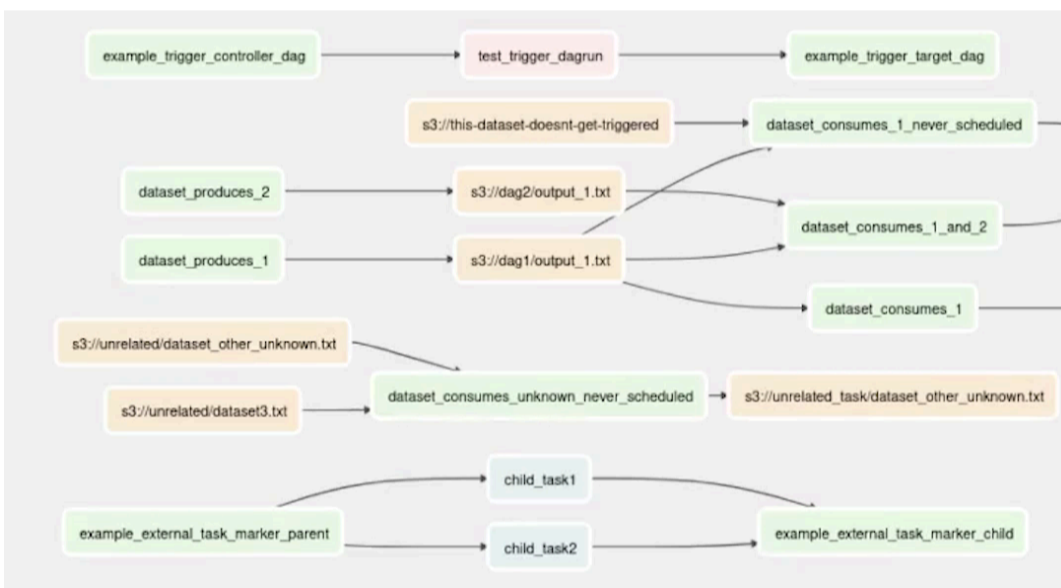
Нажимая на сам task, мы можем отфильтровывать по ID какого-либо taska и получить те же значения. Здесь они хранятся все:

State	Dag Id	Task Id	Run Id	Map Index	Logical Date	Operator	Start Date
success	user_processing	create_table	scheduled_2023-06-25T00:00:00+00:00		2023-06-25, 00:00:00	PostgresOperator	2023-06-26, 1
success	user_processing	extract_user	scheduled_2023-06-25T00:00:00+00:00		2023-06-25, 00:00:00	SimpleHttpOperator	2023-06-26, 1
failed	user_processing	process_user	scheduled_2023-06-25T00:00:00+00:00		2023-06-25, 00:00:00	PythonOperator	2023-06-26, 1
success	user_processing	is_api_available	scheduled_2023-06-25T00:00:00+00:00		2023-06-25, 00:00:00	HttpSensor	2023-06-26, 1
success	user_processing	create_table	manual_2023-06-26T13:23:17.917801+00:00		2023-06-26, 13:23:17	PostgresOperator	2023-06-26, 1
failed	user_processing	is_api_available	manual_2023-06-26T13:23:17.917801+00:00		2023-06-26, 13:23:17	HttpSensor	2023-06-26, 1
success	user_processing	extract_user	manual_2023-06-26T13:23:17.917801+00:00		2023-06-26, 13:23:17	SimpleHttpOperator	2023-06-26, 1
success	user_processing	process_user	manual_2023-06-26T13:23:17.917801+00:00		2023-06-26, 13:23:17	PythonOperator	2023-06-26, 1
success	user_processing	create_table	manual_2023-06-26T13:30:56.935640+00:00		2023-06-26, 13:30:56	PostgresOperator	2023-06-26, 1
success	user_processing	is_api_available	manual_2023-06-26T13:30:56.935640+00:00		2023-06-26, 13:30:56	HttpSensor	2023-06-26, 1

Здесь есть дополнительные поля, которых не видно (end date, длительность, в каком job'е был запущен таск, номер попытки, очереди выполнения):

State	Dag Id	Task Id	Run Id	Map Index	Logical Date	Operator	Start Date	End Date	Duration	Job Id	Queue
success	user_processing	create_table	manual_2023-06-26T13:30:56.935640+00:00	13	2023-06-26, 13:30:56	PostgresOperator	2023-06-26, 13:30:56	2023-06-26, 13:31:01	~1s	airflow_2023-06-26T13:30:56.935640+00:00	default
failed	user_processing	process_user	manual_2023-06-26T13:30:56.935640+00:00	14	2023-06-26, 13:30:56	PythonOperator	2023-06-26, 13:30:56	2023-06-26, 13:31:01	~5s	airflow_2023-06-26T13:30:56.935640+00:00	default
success	user_processing	extract_user	manual_2023-06-26T13:30:56.935640+00:00	15	2023-06-26, 13:30:56	SimpleHttpOperator	2023-06-26, 13:30:56	2023-06-26, 13:31:01	~5s	airflow_2023-06-26T13:30:56.935640+00:00	default
success	user_processing	is_api_available	manual_2023-06-26T13:30:56.935640+00:00	16	2023-06-26, 13:30:56	HttpSensor	2023-06-26, 13:30:56	2023-06-26, 13:31:01	~5s	airflow_2023-06-26T13:30:56.935640+00:00	default
success	user_processing	create_table	manual_2023-06-26T13:30:56.935640+00:00	17	2023-06-26, 13:30:56	PostgresOperator	2023-06-26, 13:30:56	2023-06-26, 13:31:01	~1s	airflow_2023-06-26T13:30:56.935640+00:00	default

DAG Dependencies – это зависимости между DAG, их можно задавать. Также здесь есть несколько зависимостей из Example (это случается, когда DAG проверяет, не появилось ли что-то на S3 и т.д.):



Заходим в **Admin**. Здесь хранятся переменные, где можно добавлять ключ переменной, значение. Они хранятся в зашифрованном виде Metastore, но при этом вы можете иметь к ним доступ в коде. **Какие-то более чувствительные значения лучше хранить в другом месте:**

Add Variable

Key *

Val

Description

Connections мы уже прописывали на user processing. Поскольку они хранятся в одном месте, желательно описывать description:

Conn Id	Conn Type	Description	Host	Port	Is Encrypted	Is Extra Encrypted
postgres	postgres		postgres	5432	False	False
user_api	http		https://randomuser.me/		False	False

Провайдеры – это встроенные связи AirFlow с внешними сервисами. Их довольно много, также можно дописывать СВОИ:

Providers

Package Name	Version	Description
apache-airflow-providers-amazon	8.1.0	Amazon integration (including Amazon Web Services (AWS)).
apache-airflow-providers-celery	3.2.0	Celery
apache-airflow-providers-cncf-kubernetes	7.0.0	Kubernetes
apache-airflow-providers-common-sql	1.5.1	Common SQL Provider
apache-airflow-providers-docker	3.7.0	Docker
apache-airflow-providers-elasticsearch	4.5.0	Elasticsearch
apache-airflow-providers-ftp	3.4.1	File Transfer Protocol (FTP)
apache-airflow-providers-google	10.1.1	Google services including: - Google Ads - Google Cloud (GCP) - Google Firebase - Google LevelDB - Google Marketing Platform - Google Workspace (formerly Google Suite)
apache-airflow-providers-grpc	3.2.0	gRPC
apache-airflow-providers-hashicorp	3.4.0	Hashicorp including Hashicorp Vault
https://airflow.apache.org/docs/apache-airflow-providers-celery/3.2.0/	4.4.1	Hypertext Transfer Protocol (HTTP)

List Pool по умолчанию – 128:

<input type="checkbox"/>	Pool	Slots	Running Slots	Queued Slots	Scheduled Slots
<input type="checkbox"/>	default_pool	128	0	0	0

List XComs. Здесь хранятся XCom'ы:

<input type="checkbox"/>	Key	Value	Timestamp	Dag Id	Task Id
<input type="checkbox"/>	return_value	1	2023-06-26, 16:09:43	example_task_group	section_
<input type="checkbox"/>	return_value	1	2023-06-26, 16:09:45	example_task_group	section_
<input type="checkbox"/>	return_value	Hello, I am a value!	2023-06-26, 16:11:24	xcom_edu_dag	downloa
<input type="checkbox"/>	return_value	{'results': [{'gender': 'male', 'name': {'title': 'Mr', 'first': 'Diether', 'last': 'Hansmann'}, 'location': {'street': {'number': 1126, 'name': 'Tannenweg'}, 'city': 'Laage', 'state': 'Baden-Württemberg', 'country': 'Germany', 'postcode': 35022, 'coordinates': {'latitude': '-50.1428', 'longitude': '-114.3762'}, 'timezone': {'offset': '-6:00', 'description': 'Central Time (US & Canada), Mexico City}}, 'email': 'diether.hansmann@example.com', 'login': {'uid': '2431e704-08d4-4c74-8de7-2d93230cc08c', 'username': 'whitekoala218', 'password': 'smoking', 'salt': 'aGZqXDsX', 'md5': '0d626c8b22816331767d09db850ed3b2', 'sha1': '081743b9fddc6c120545e7ae16a065a262a2b2c2', 'sha256': '80924af30dc5005ad9d5d7af102b7795b8e9f1f643c2cd523351b8a13ba4cbde'}, 'dob': {'date': '1993-06-23T10:18:04.750Z', 'age': 30}, 'registered': {'date': '2014-01-29T20:38:52.250Z', 'age': 9}, 'phone': '0120-8705790', 'cell': '0178-7231547', 'id': {'name': 'SVNR', 'value': '74 230693 H 167'}, 'picture':	2023-06-26, 15:27:03	user_processing	extract_

Вы можете посмотреть все XCom'ы задач, которые были помещены в Metastore. Они все хранятся в Metastore, отсюда ограничение XCom в зависимости от типа базы, используемой Metastore.

Вверху указано **время UTC**. Это тоже важно для конфигурации Web-view.

Мы познакомились с Web-view и важными частями. Теперь мы знаем следующее:

- какие виды отображения DAG позволяют решать разные задачи;
- какие виды отображения DAG позволяют нам видеть, где выполнение задачи занимают много времени, а где много файлов.

Как вам урок?



Изучил, далее >