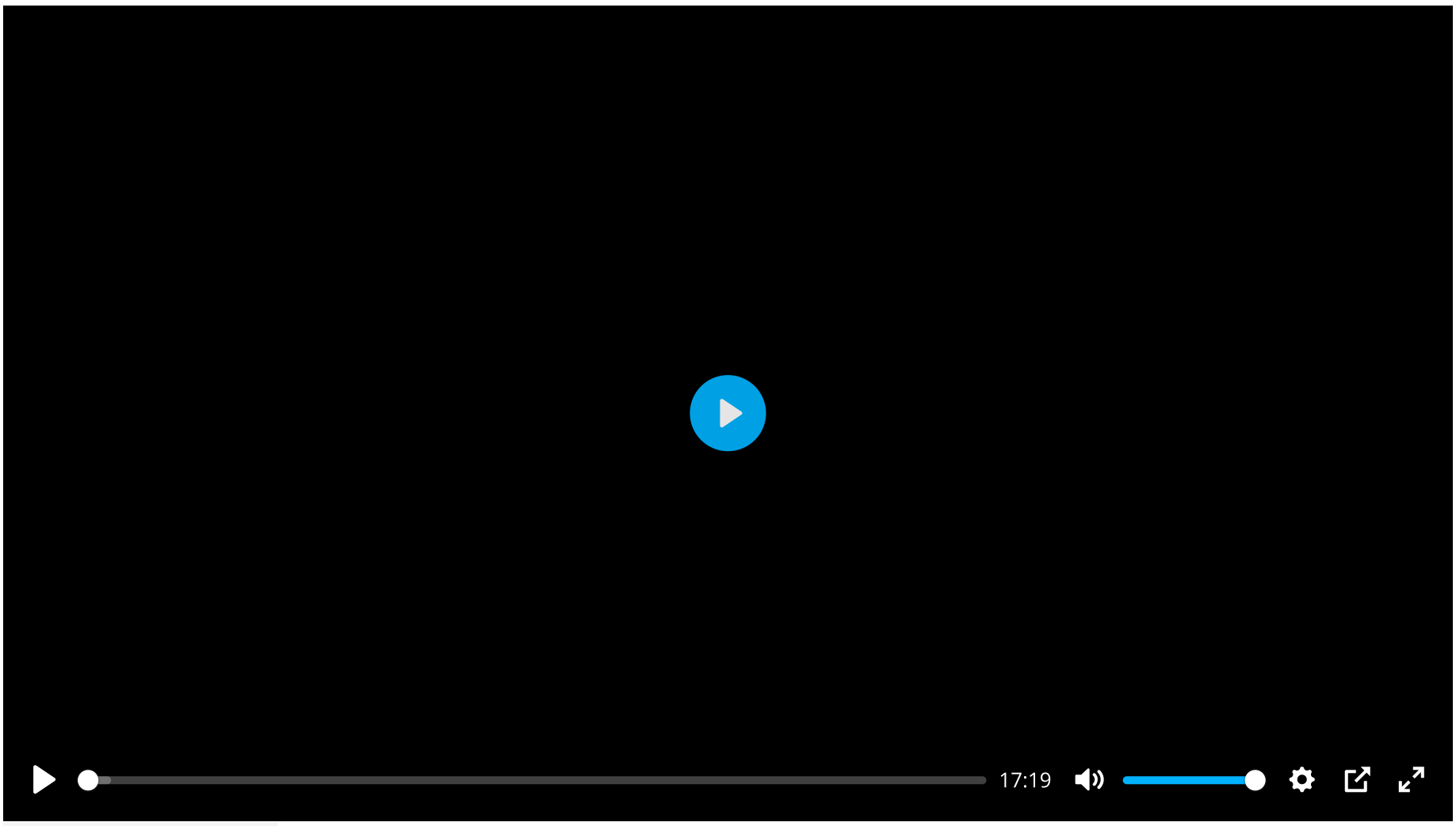
8.4 Работа с компонентами Airflow



Презентация к уроку 8.4.1

Текстовая расшифровка видео:

КОМПОНЕНТЫ AIRFLOW. WEBVIEW

План:

- WebView.

WebView

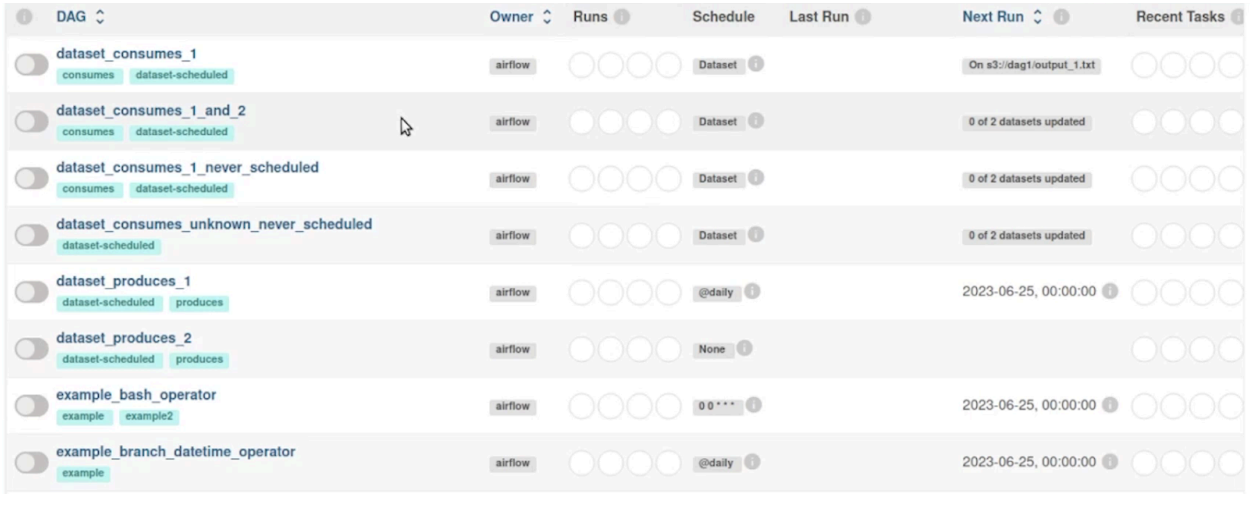
WebView – это компонент, через который осуществляется управление инстансом Airflow. Чтобы его открыть, необходимо ввести в браузере следующее:

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.

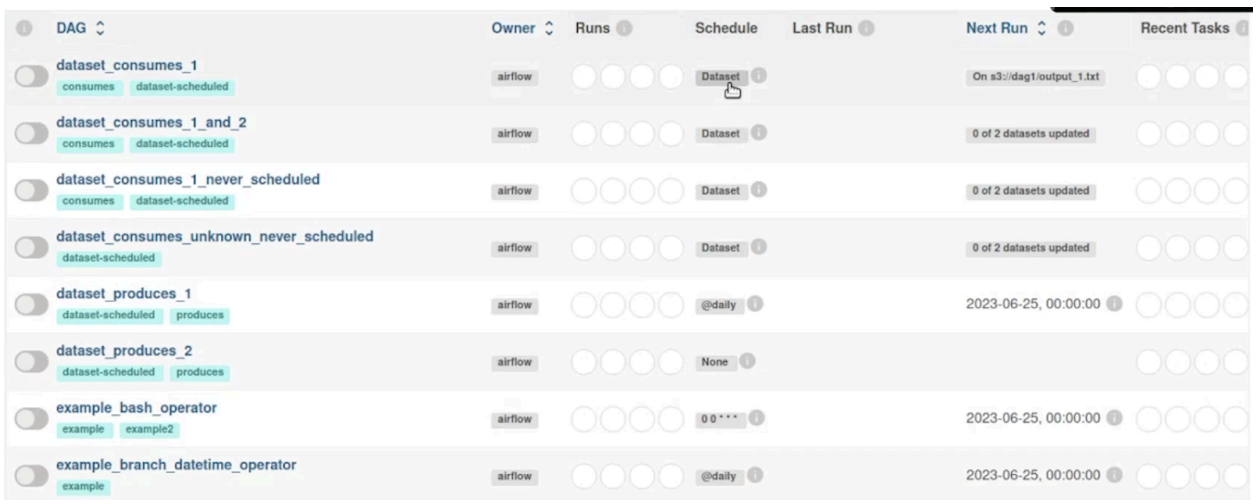
С запущенным инстансом через docker compose app открываем браузер и видим web-view. Здесь хранятся все DAGs (на данный момент – DAGs-премьеры):



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:

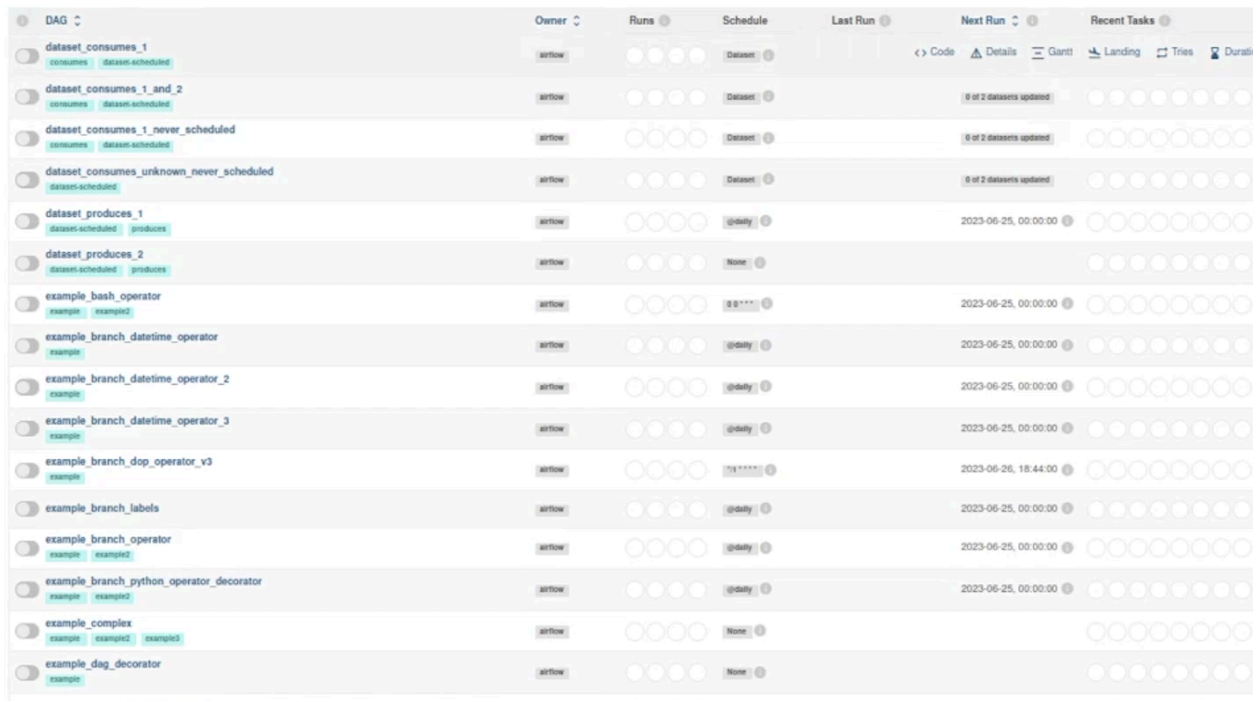
Здесь также указывается интервал schedule, то есть частота повтора DAG. Если none, то он не повторяется. Остальные значения означают, с какой периодичностью мы его запускаем:



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

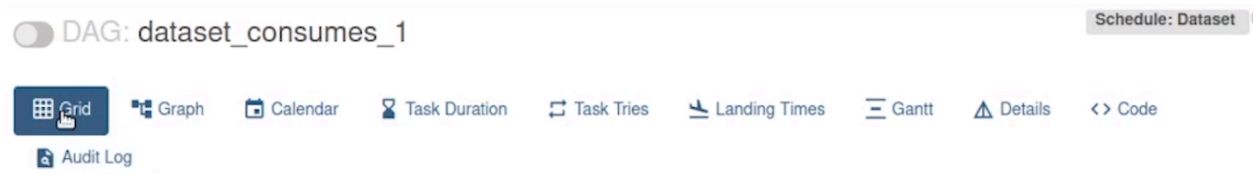
Next Run – планировщик следующего запуска.

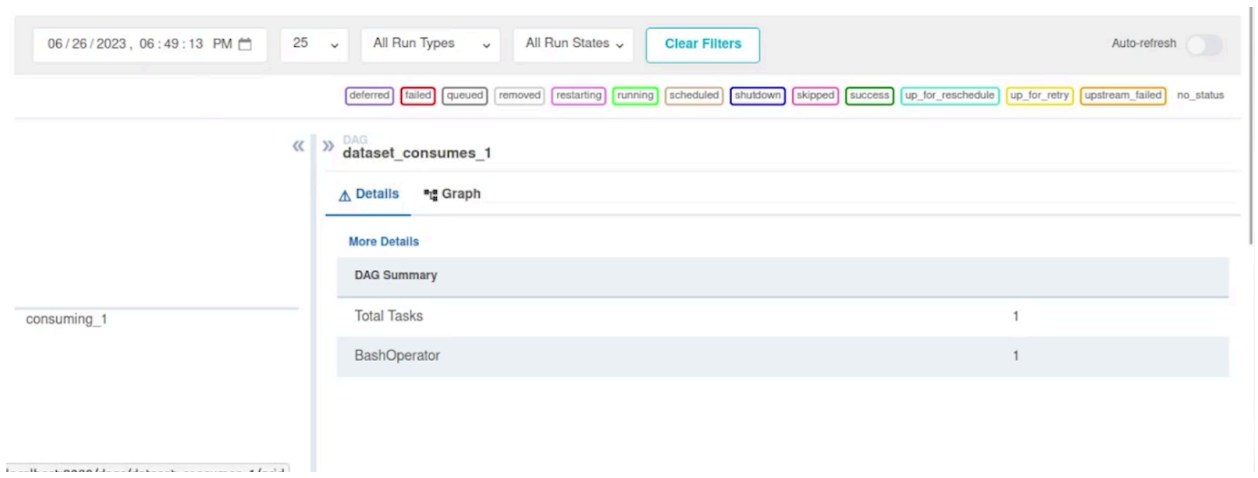
Recent Tasks – это таски, которые исполнялись. Каждый кружочек обозначает статус и количество выполнений.
Это действие, которое можно производить над DAG'ом. Вы можете запустить его вручную отсюда:



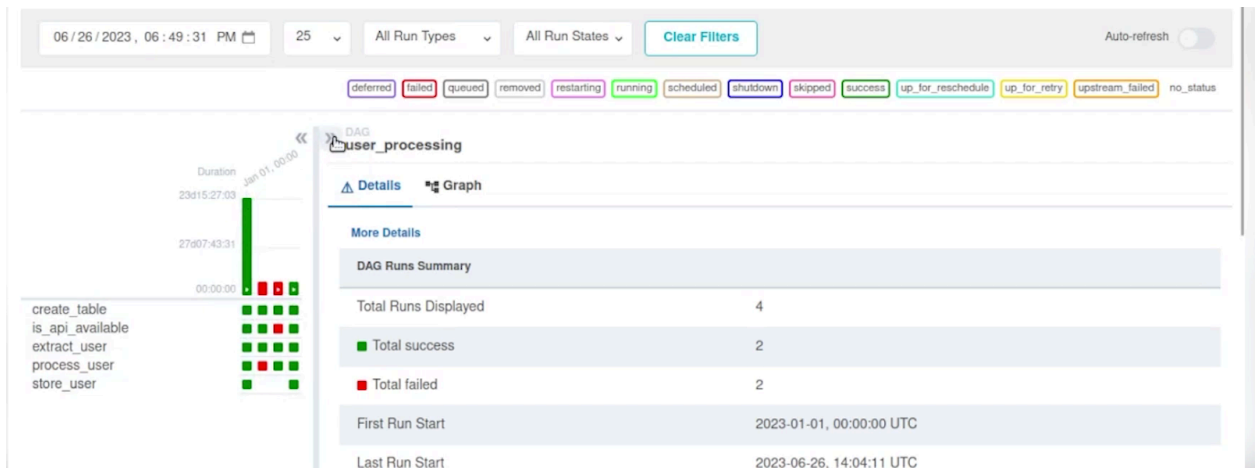
Также вы можете выбрать любое из отображений DAG в этом разделе. Чтобы что-то посмотреть, вам не обязательно заходить в сам DAG.

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

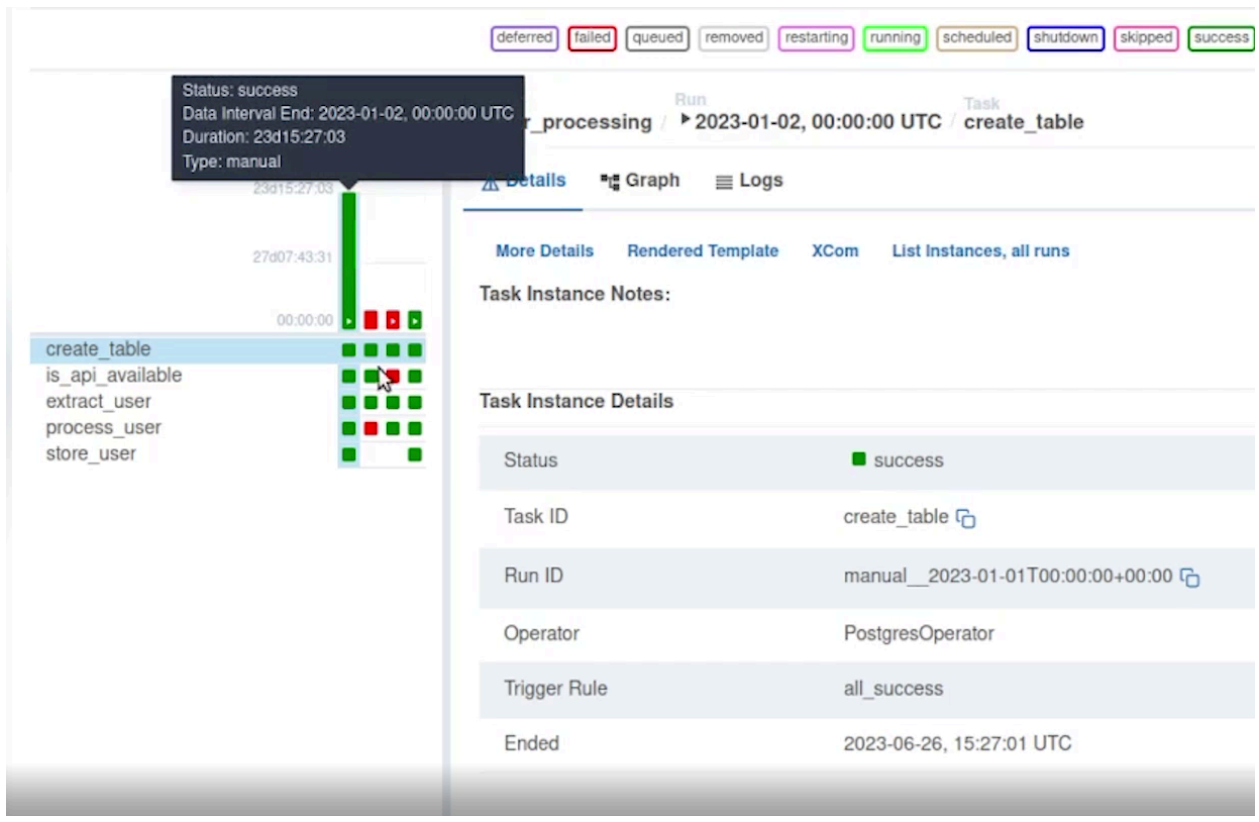




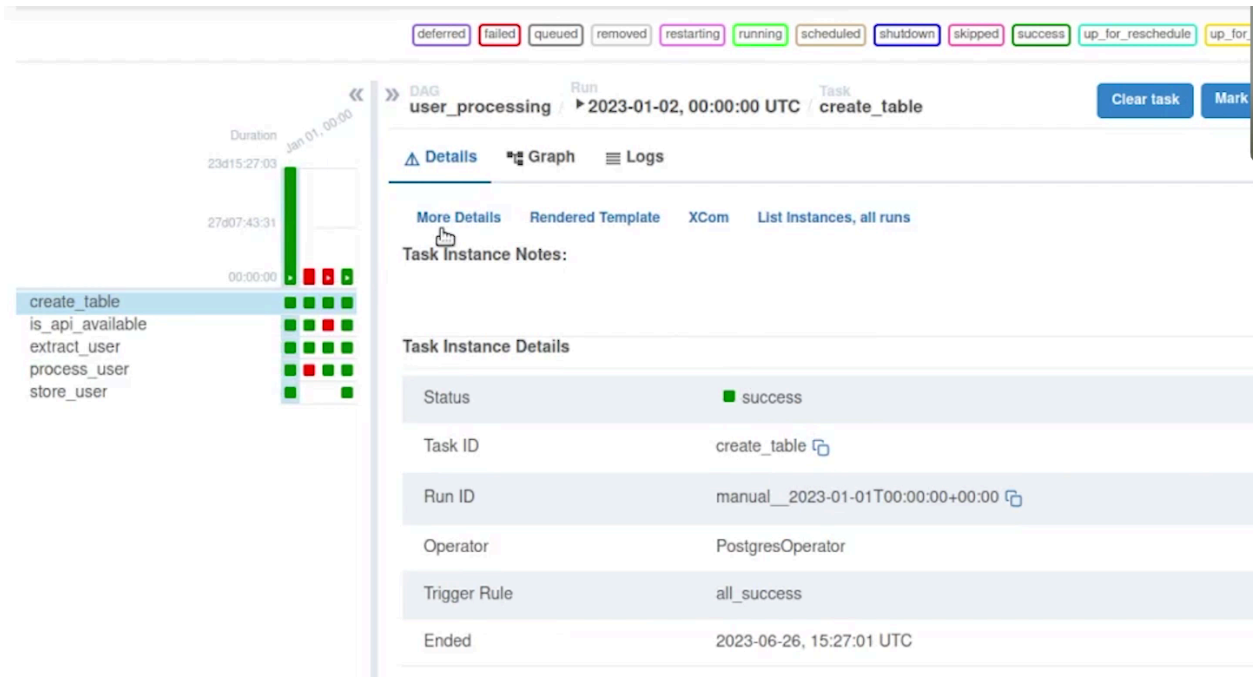
Это **Grid View** – вид-сетка. Посмотрим User Processing. Здесь вид-сетка обозначает следующее:



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



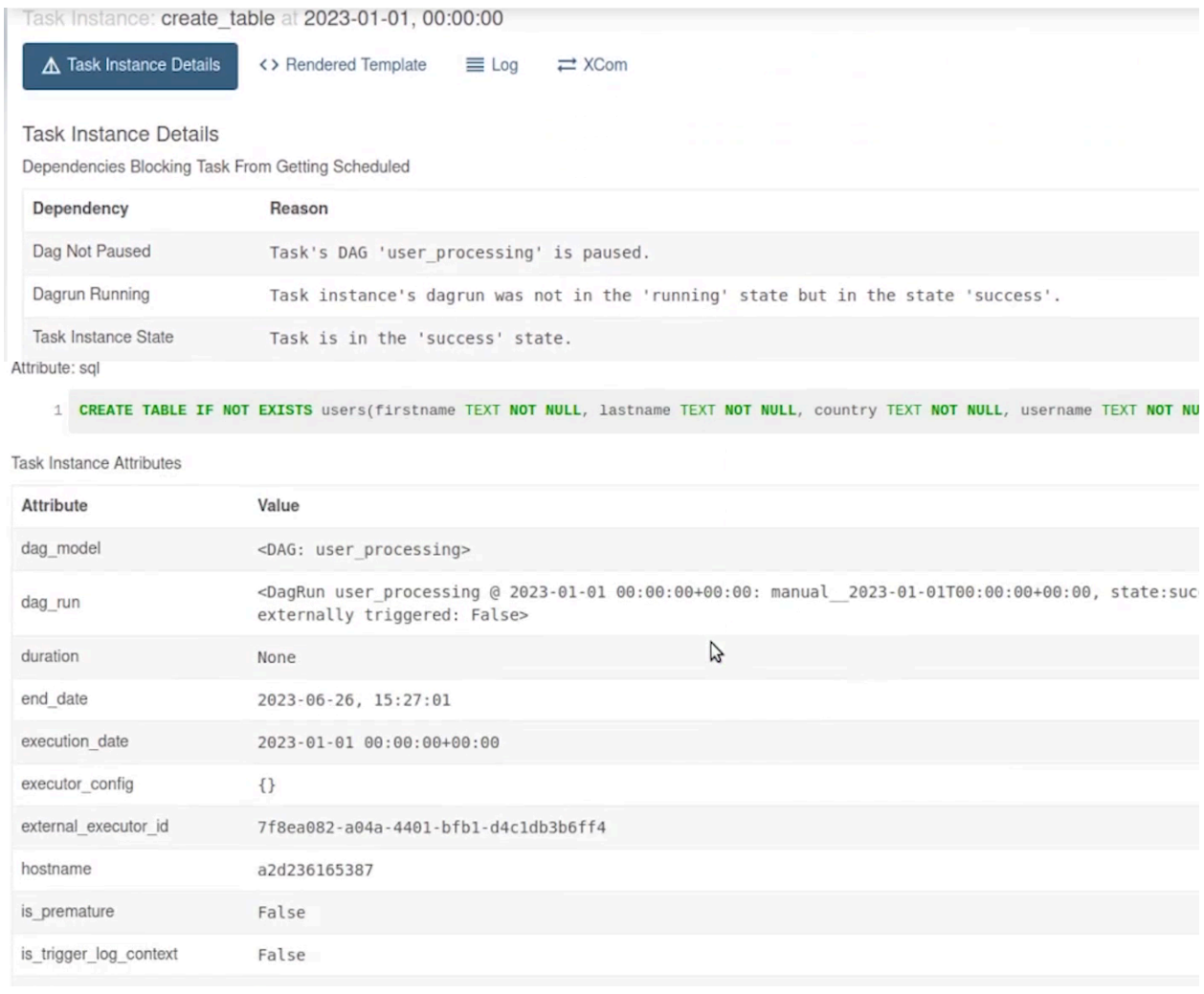
Все это можно спрятать, также можно посмотреть целиком для DAG, то есть для тасков и table:



Мы видим следующее:

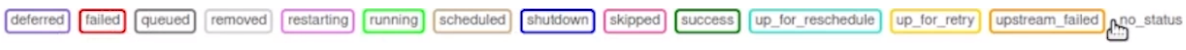
- Статус таска;
- Task ID;
- Run ID, где manual означает, что он был запущен вручную;
- Отображение оператора;
- Trigger Rule (почему он запустится; когда он был завершен).

Здесь располагаются все детали по данному таску:



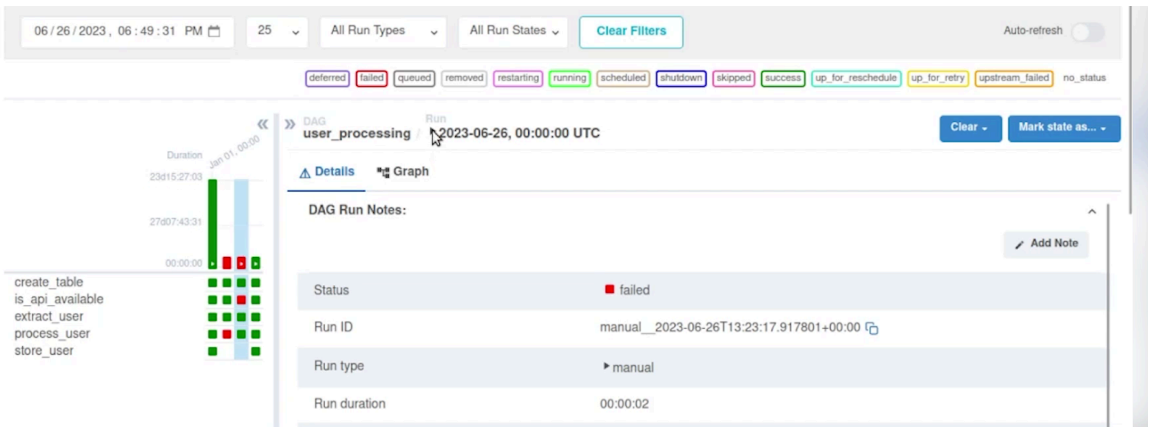
- Из чего он состоит;
- Какие у него есть свойства.

Это статусы тасков:

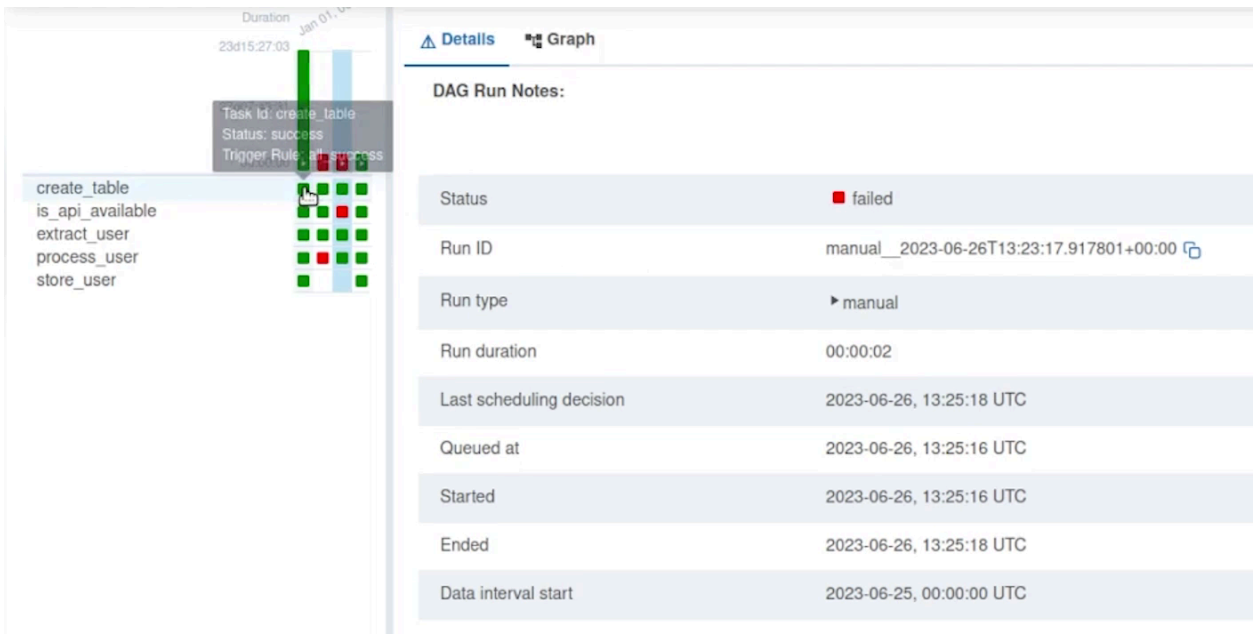


Их можно увидеть, как в Gridview, так в другом формате. В Gridview тоже доступны такие опции.

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

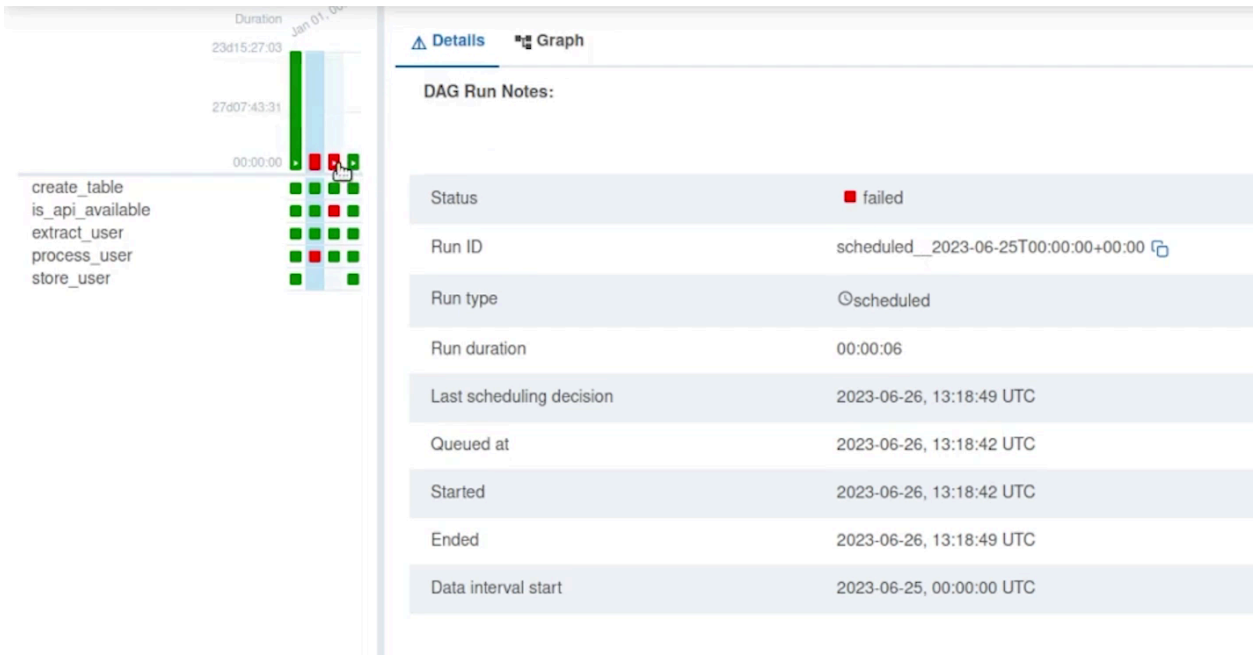


DAG Run-объект – это объект отдельного запуска DAG. По префиксу вначале мы видим «manual», то есть был совершен ручной запуск (или из расписания):



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

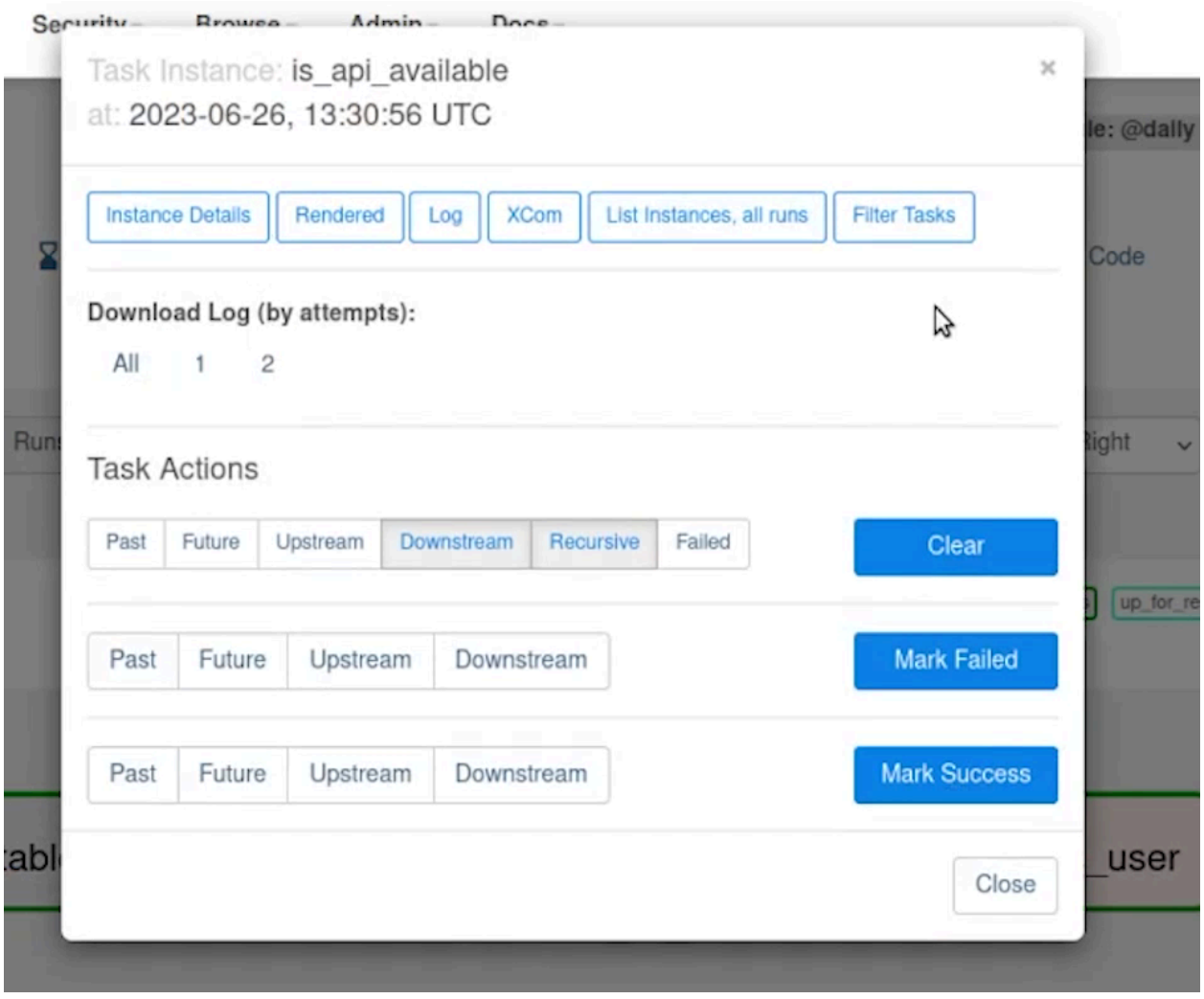
В данном случае указан scheduled:



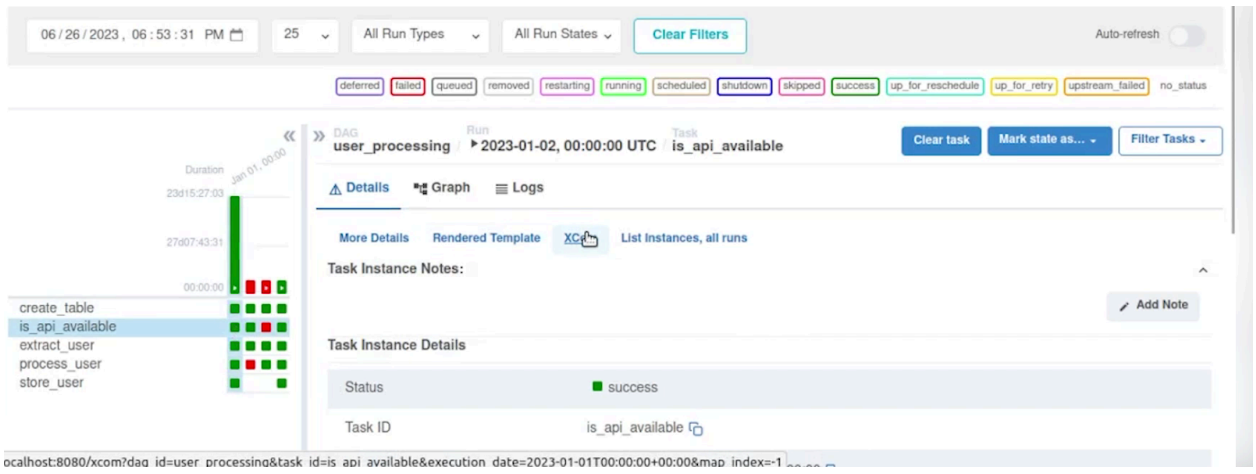
Также мы видим все параметры времени:

Перейдем в граф View:

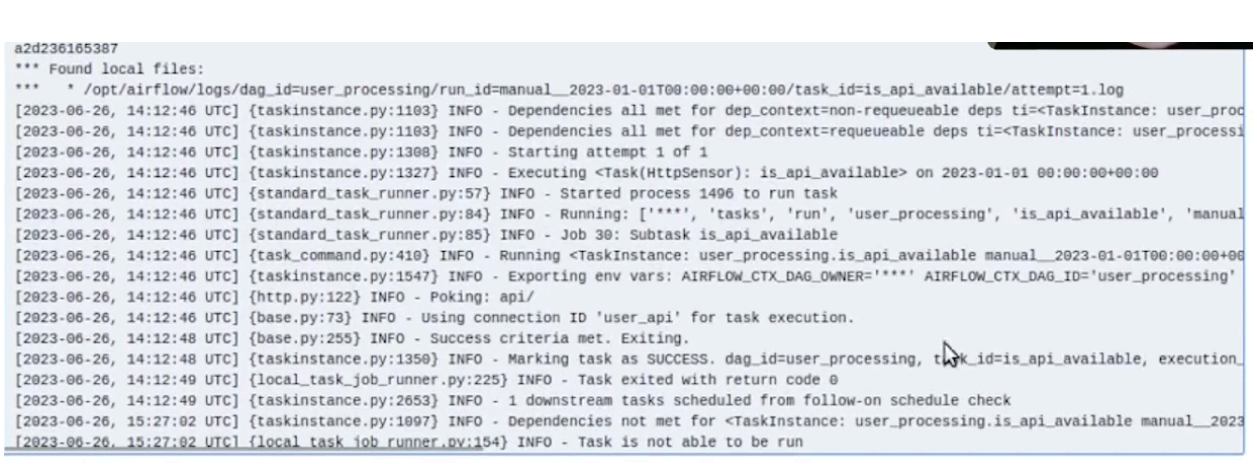
Граф View отображает одновременно один DAG Run. Он очень удобен, чтобы посмотреть информацию по каждому таску. Здесь можно выбирать какой-то конкретный Run. Удобно и то, что мы можем выбрать конкретный таск и посмотреть в нем свойства, или очистить его:



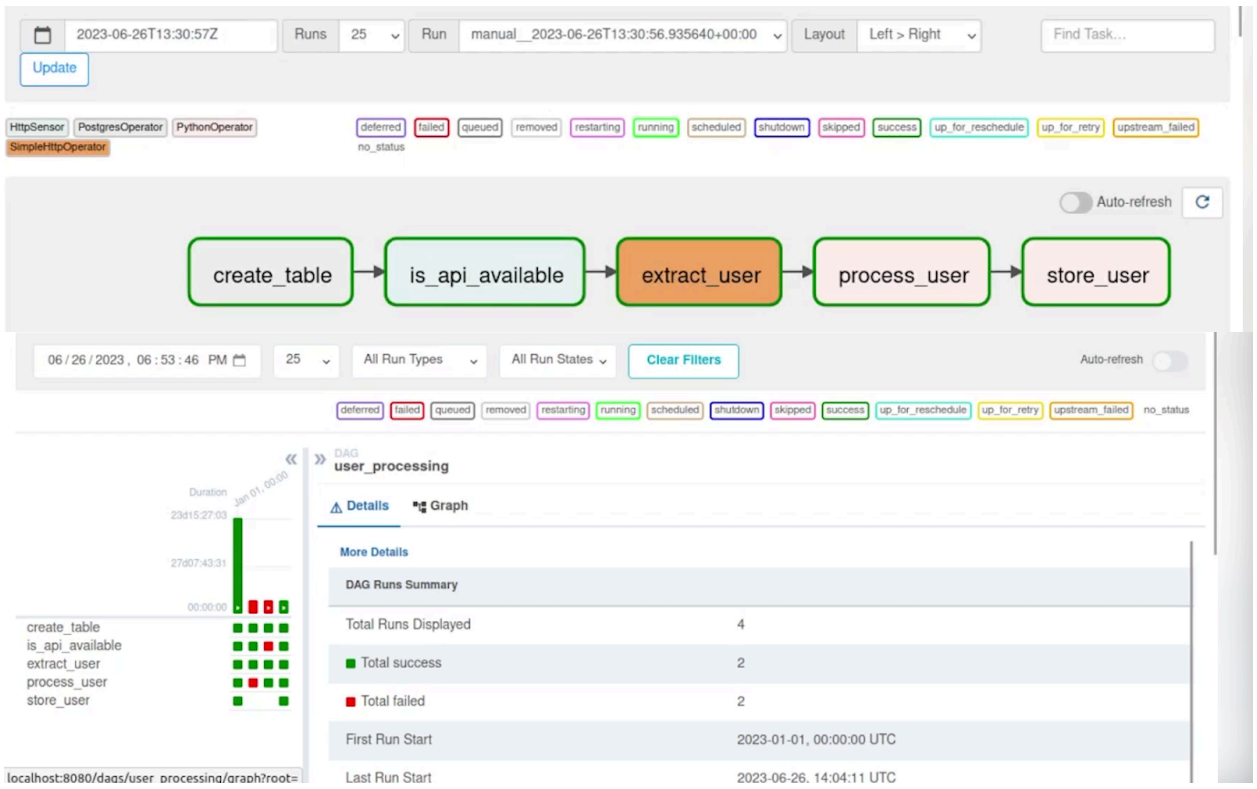
Все эти параметры дублируются. Вы можете смотреть их в **вид-сетка**, если вам так удобнее:



Или в логе:

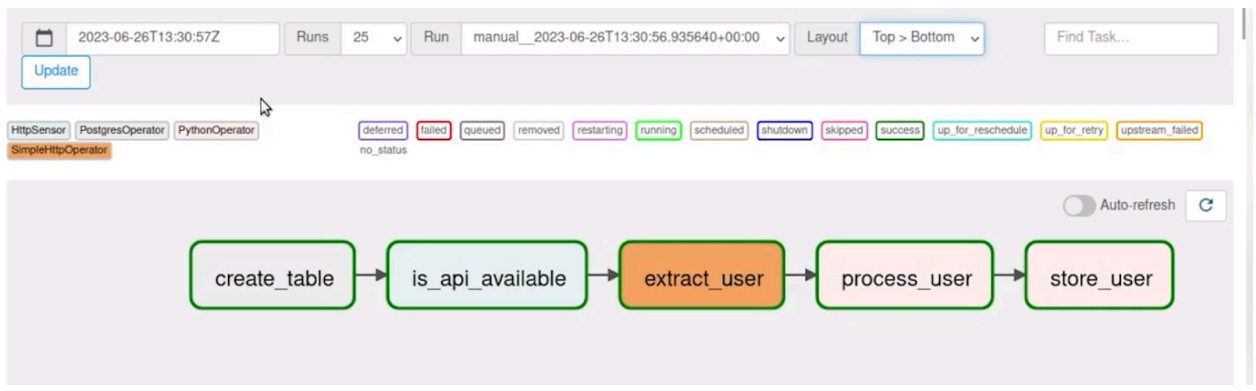


Или в графе:

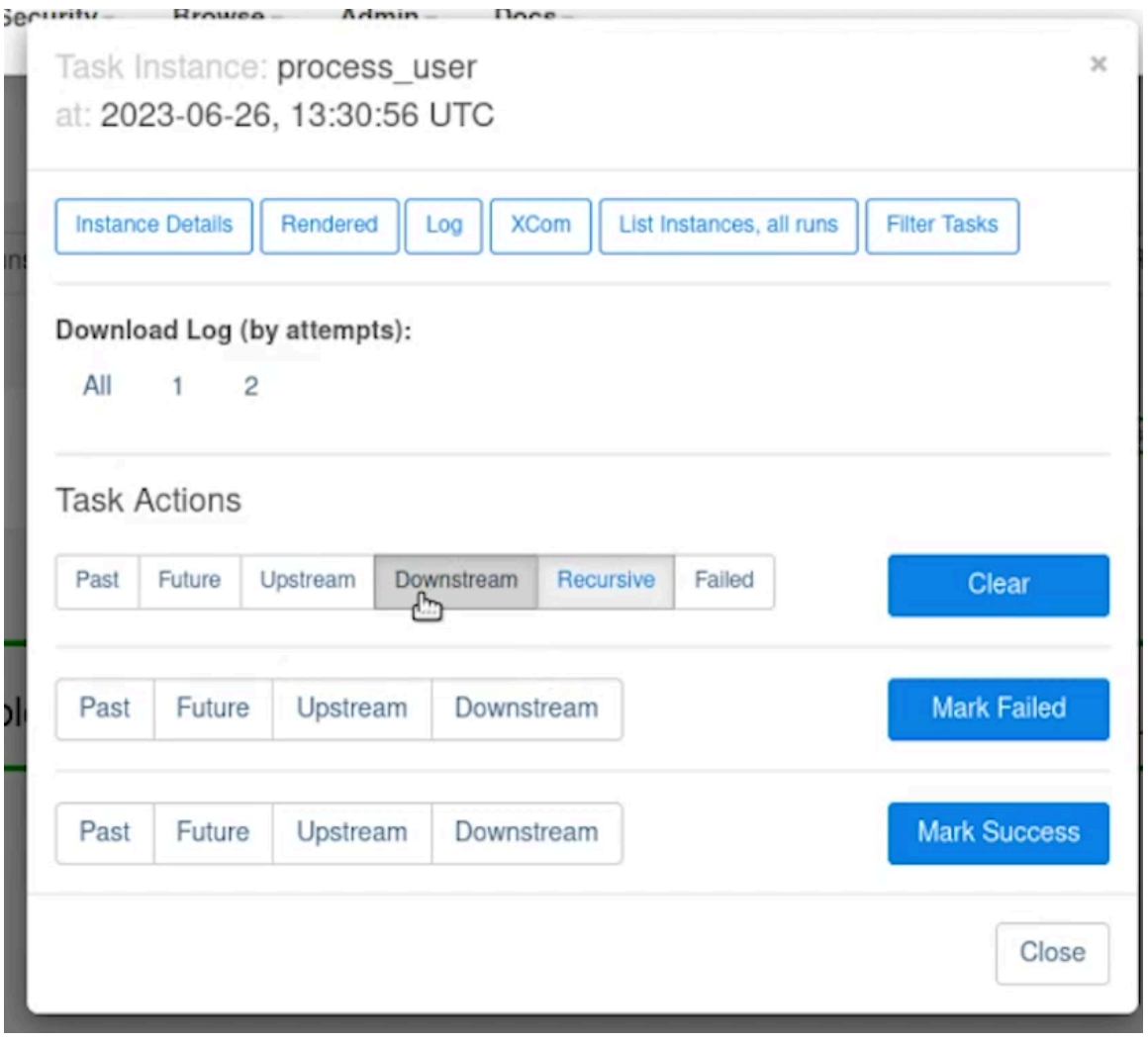


Однако разница все же присутствует:

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



Здесь мы можем **очищать**:

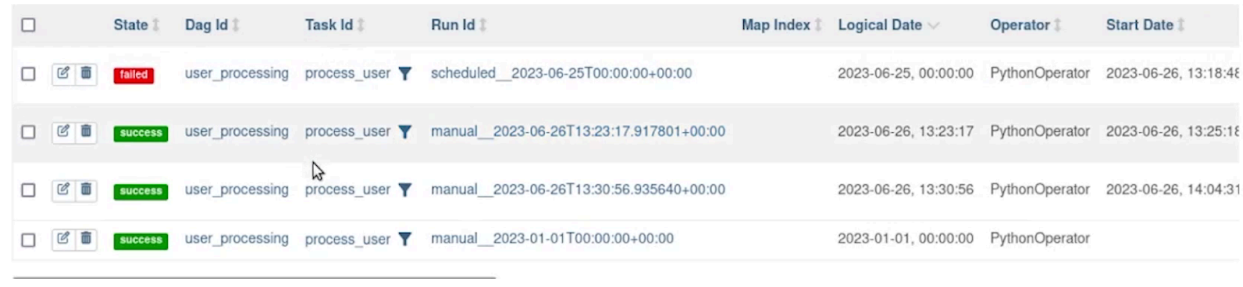


Правило: очищаются все последующие. Однако правило можно поменять. Так, очищается вся информация по таску и сами Task Run’ы. По желанию их можно отметить Failed или Success.

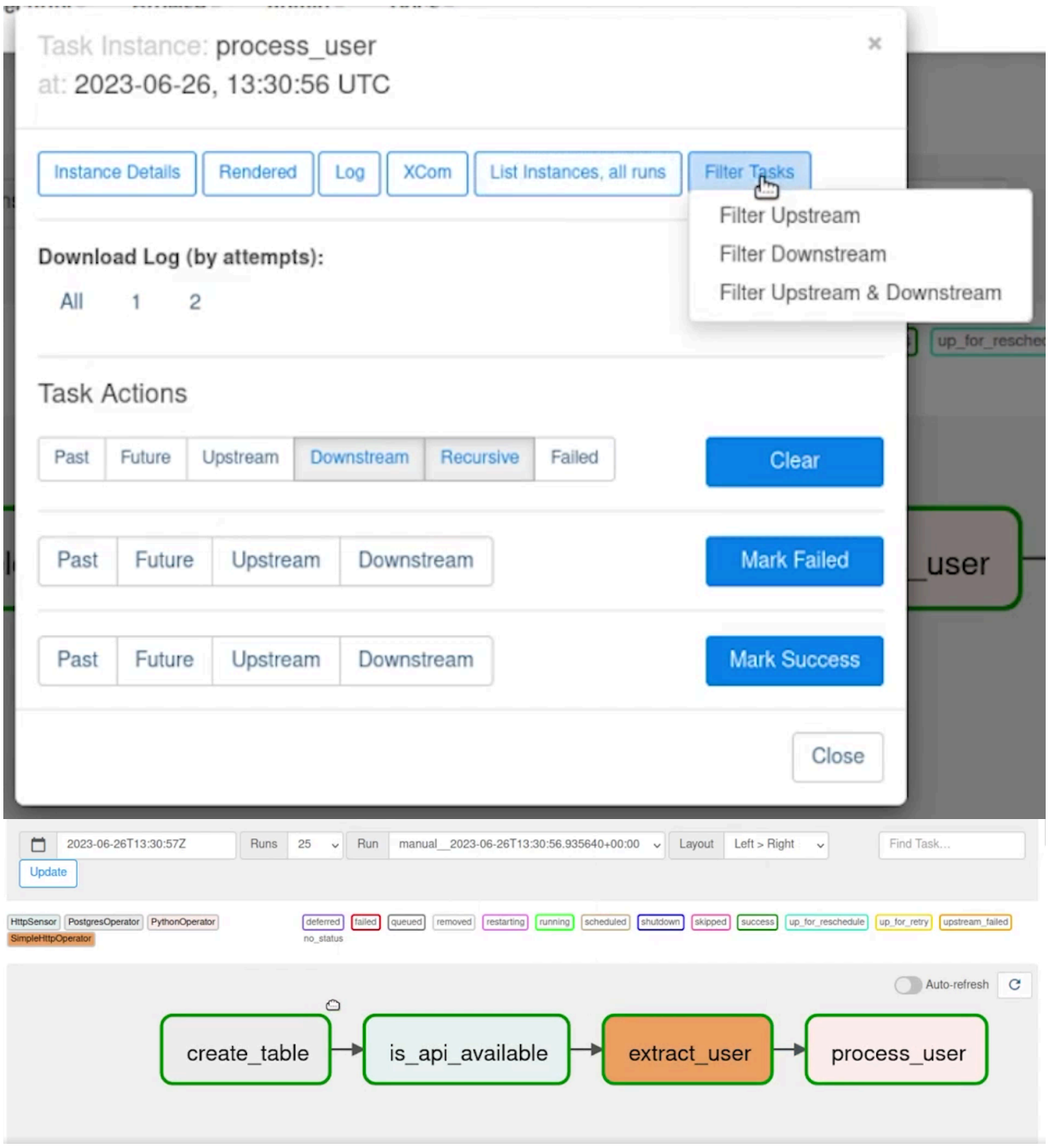
Здесь хранятся все данные о Task instance, как и в Gridview:

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

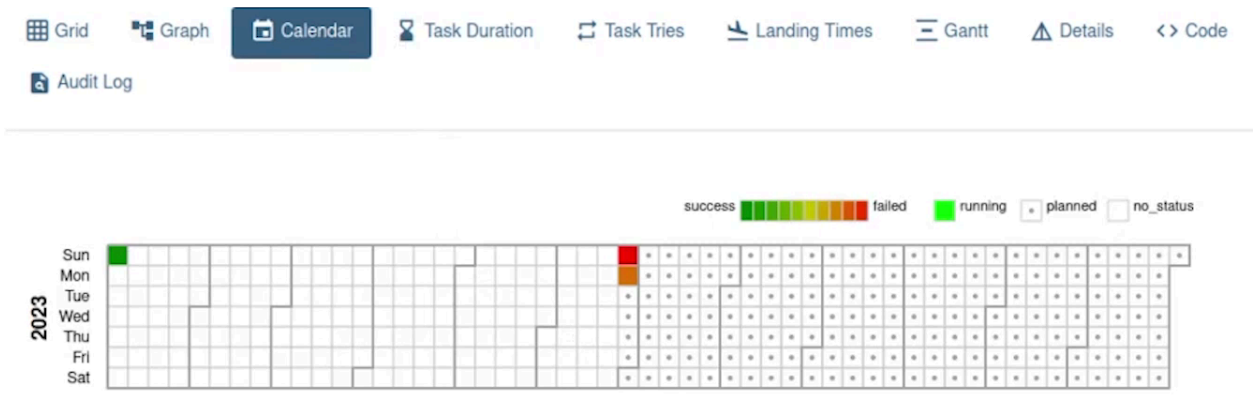
Это еще один способ посмотреть все таск-инстансы именно по этому таску:



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

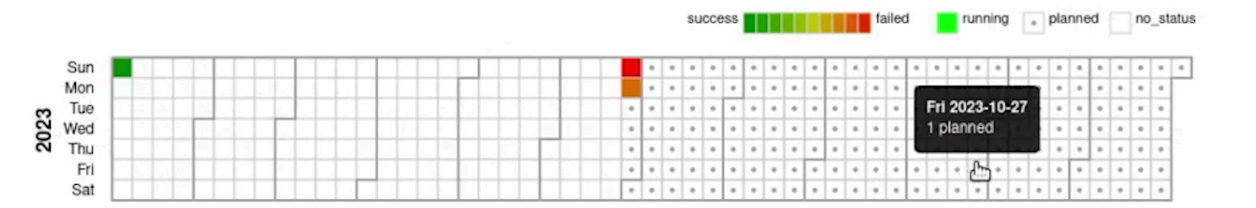


Вид-календарь.

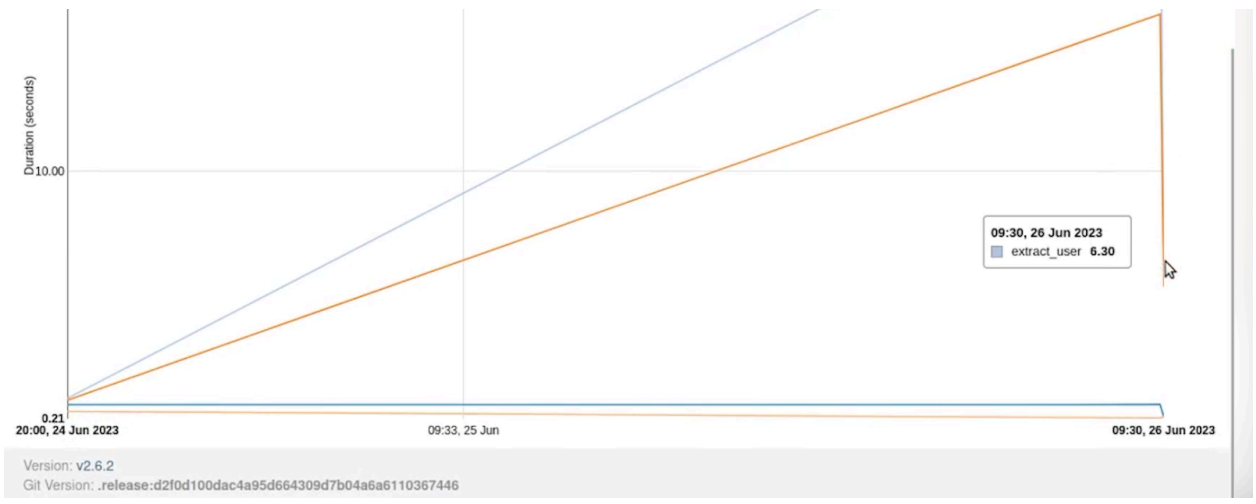


Вид-календарь удобен тем, что можно посмотреть по календарю в какие дни особенно бывает просадка или много падений задач. Например, если это выходные, то можно предусмотреть. Чем квадратик более зеленый, тем больший процент задач выполнялся успешно.

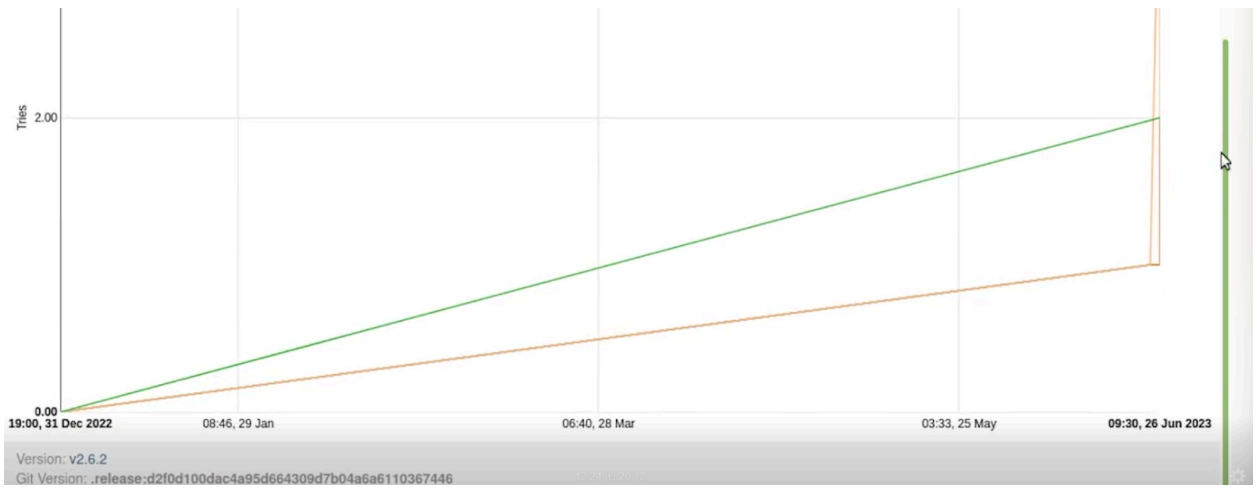
Точки в середине квадратов – это запланированные на будущее задачи:



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:

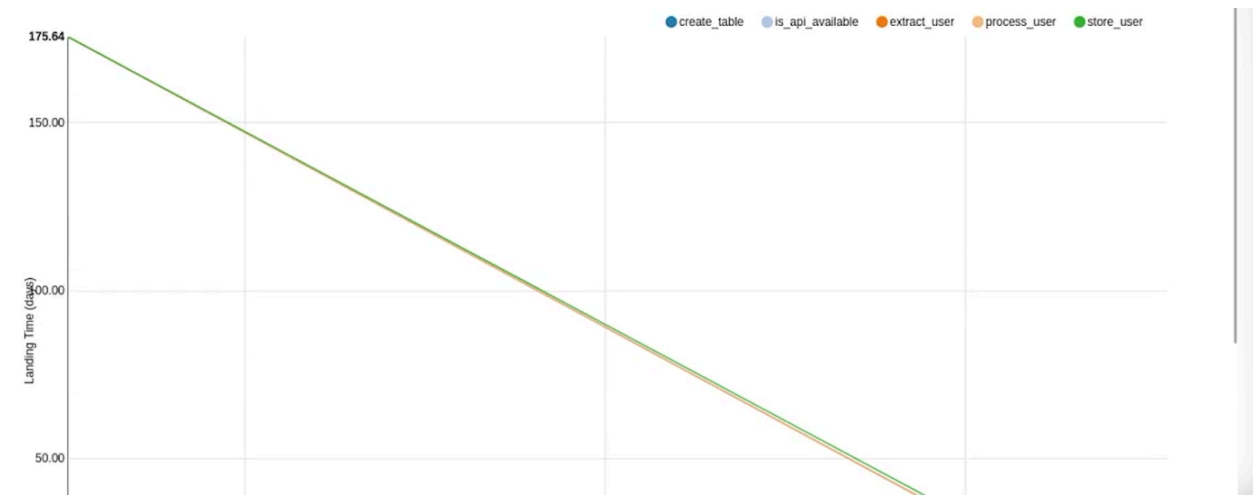
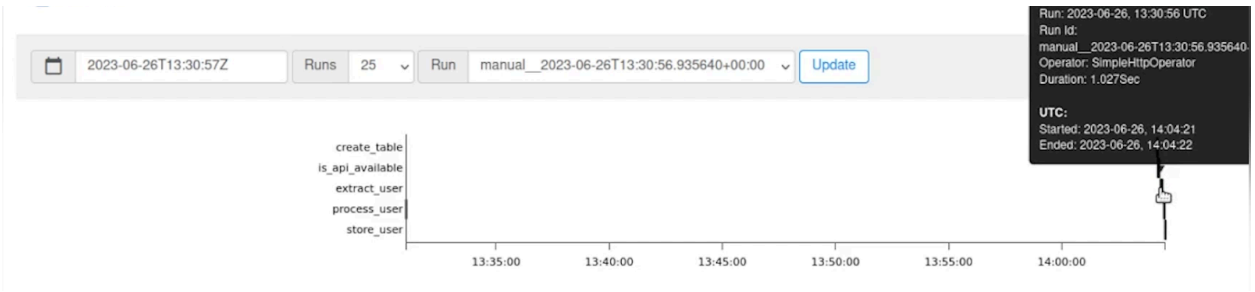


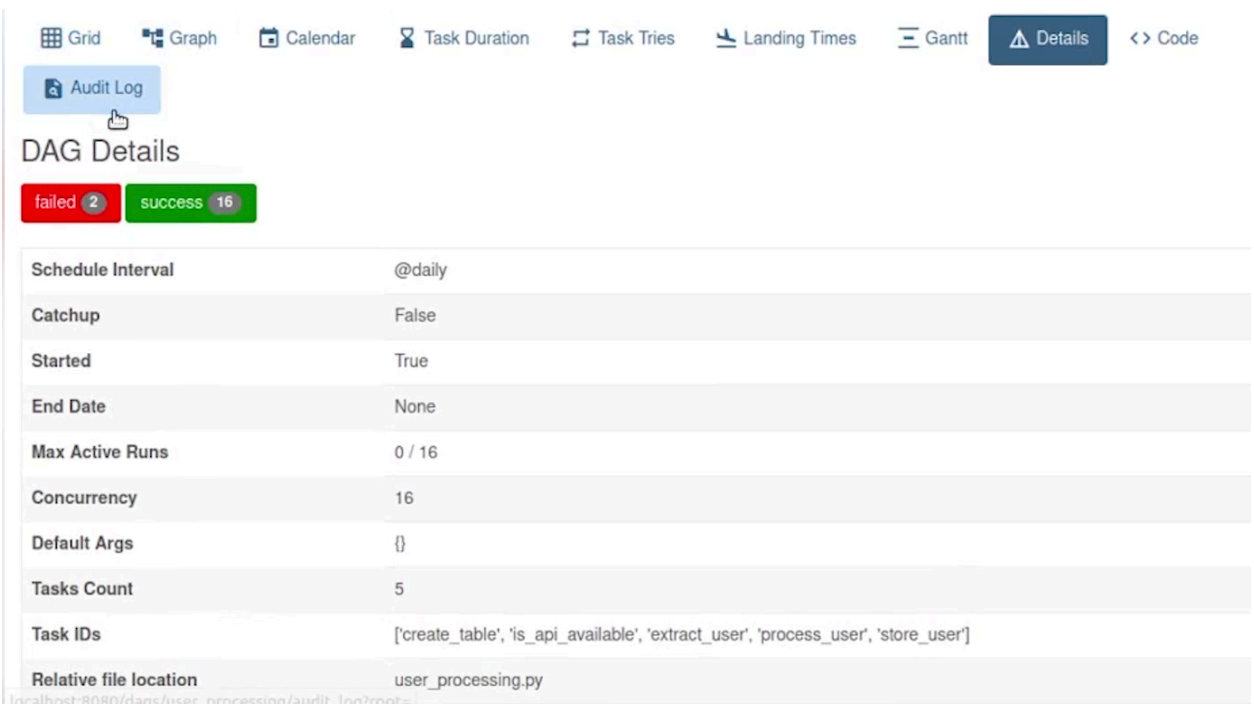
Диаграмма Gantt показывает длительность выполнения каждого таска. Если видим очень длинную зеленую полоску, то это значит, что какой-то из тасков замедляет выполнение DAG. Это так называемое «Бутылочное горлышко», на Landing Times это тоже можно увидеть:



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:

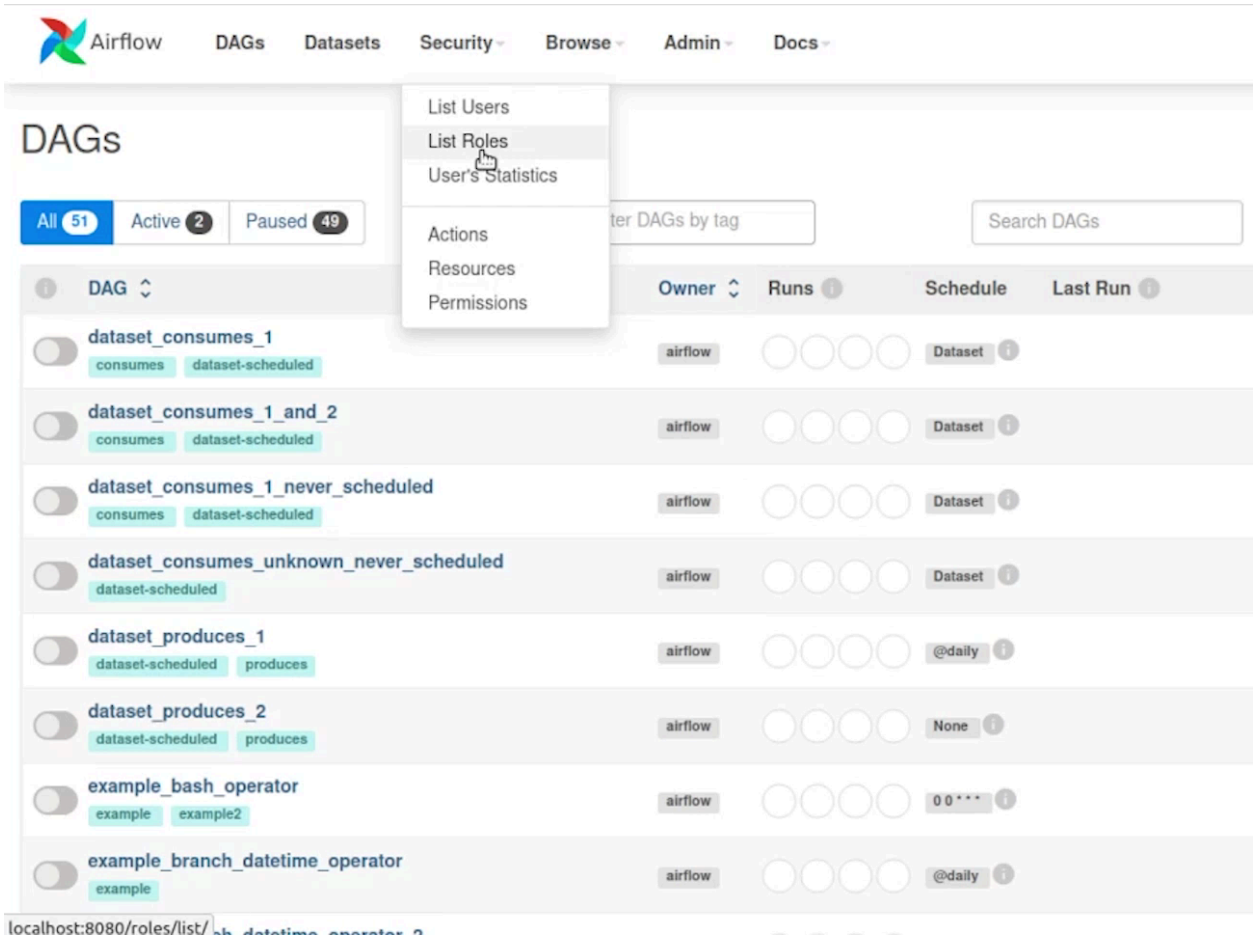


Code. Здесь удобно смотреть внеслись ли изменения, которые вы добавили в файл в директории DAGs и др. Это более быстрый доступ:

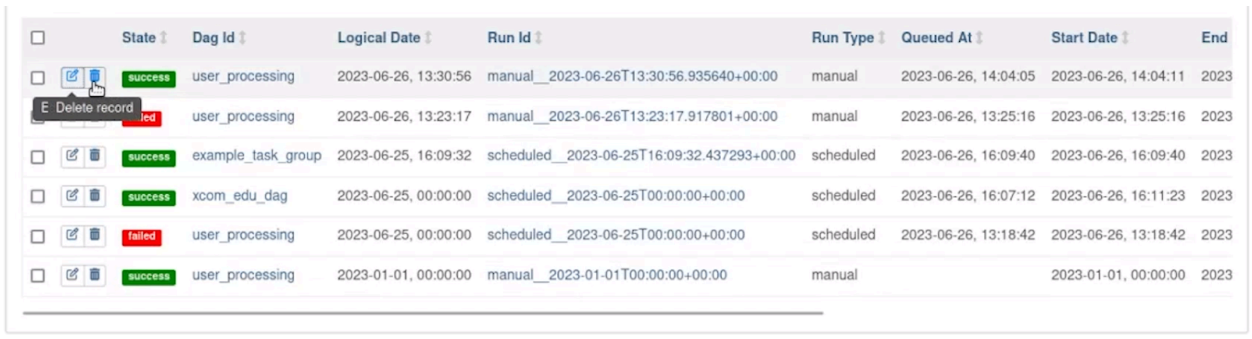
```
    processed_user = json_normalize({ 'firstname': user['name']['first'],
                                      'lastname': user['name']['last'],
                                      'country': user['location']['country'],
                                      'username': user['login']['username'],
                                      'password': user['login']['password'],
                                      'email': user['email']})
    processed_user.to_csv('/tmp/processed_user.csv', index=None, header=False)

def _store_user():
    hook = PostgresHook(postgres_conn_id='postgres',)
    hook.copy_expert(sql="COPY users FROM stdin WITH DELIMITER as ',' ", filename='/tmp/processed_user.csv')


with DAG('user_processing', start_date=datetime(2023,1,1), schedule_interval='@daily', catchup=False) as dag:


    create_table = PostgresOperator(task_id='create_table', postgres_conn_id='postgres',
                                    sql='''CREATE TABLE IF NOT EXISTS users(firstname TEXT NOT NULL, lastname TEXT NOT NULL,

    is_api_available = HttpSensor(task_id='is_api_available', http_conn_id='user_api', endpoint='api/')

    extract_user = SimpleHttpOperator(task_id='extract_user', http_conn_id='user_api', endpoint='api/', method='GET',
                                      response_filter=lambda response: json.loads(response.text))

    process_user = PythonOperator(task_id='process_user', python_callable=_process_user, )

    store_user = PythonOperator(task_id='store_user', python_callable=_store_user)


create_table >> is_api_available >> extract_user >> process_user >> store_user
```

Если мы нажмем на «**корзинку**», 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:

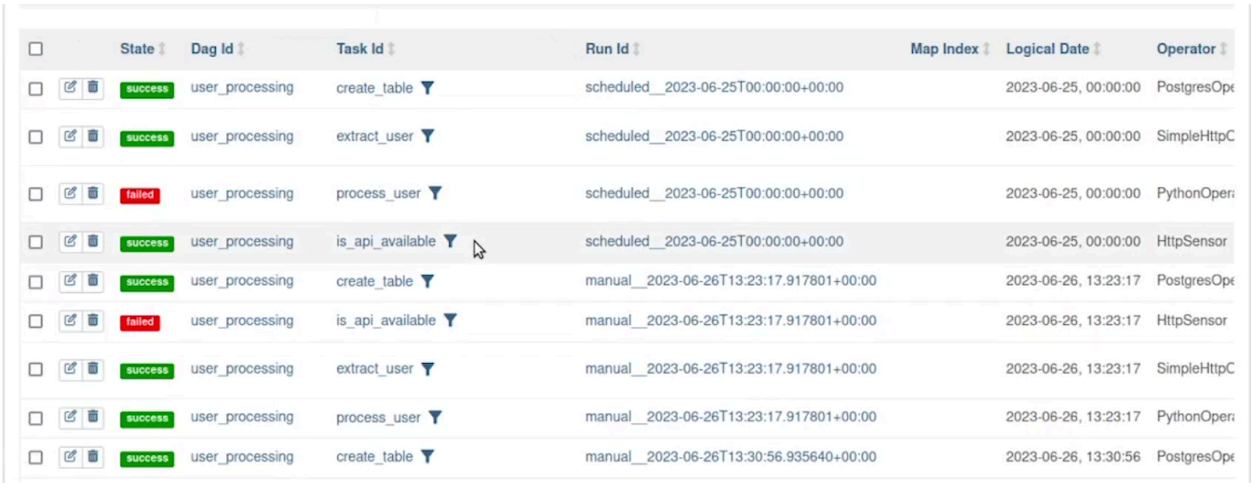


DAG Runs – это инстансы запуска DAG, то есть конкретные запуски. Их можно посмотреть по полям, также можно удалить, если что-то пошло не так (изменения, которые вносил DAG удаляться не будут):



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

Task instance – это те самые таски, которые отрабатывают:

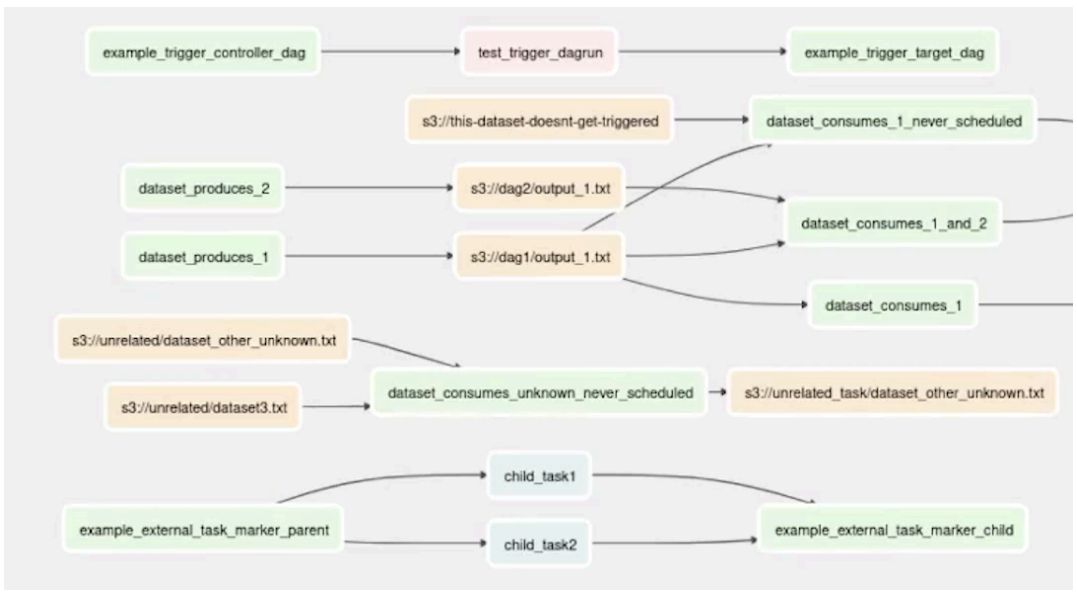


Нажимая на сам таск, мы можем отфильтровывать по ID какого-либо таска и получить те же значения. Здесь они хранятся все:

	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):

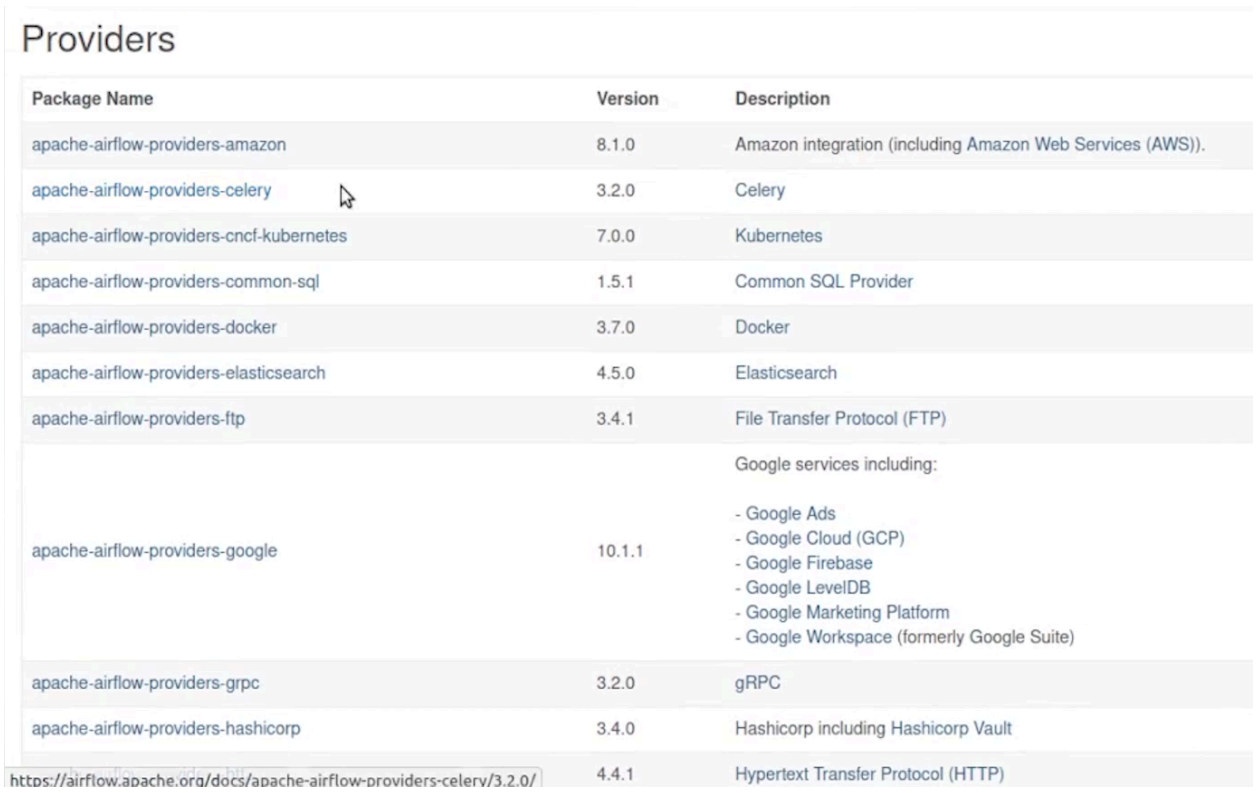
DAG Dependencies – это зависимости между DAG, их можно задавать. Также здесь есть несколько зависимостей из Example (это случается, когда DAG проверяет, не появилось ли что-то на S3 и т.д.):



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:**

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

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



Providers

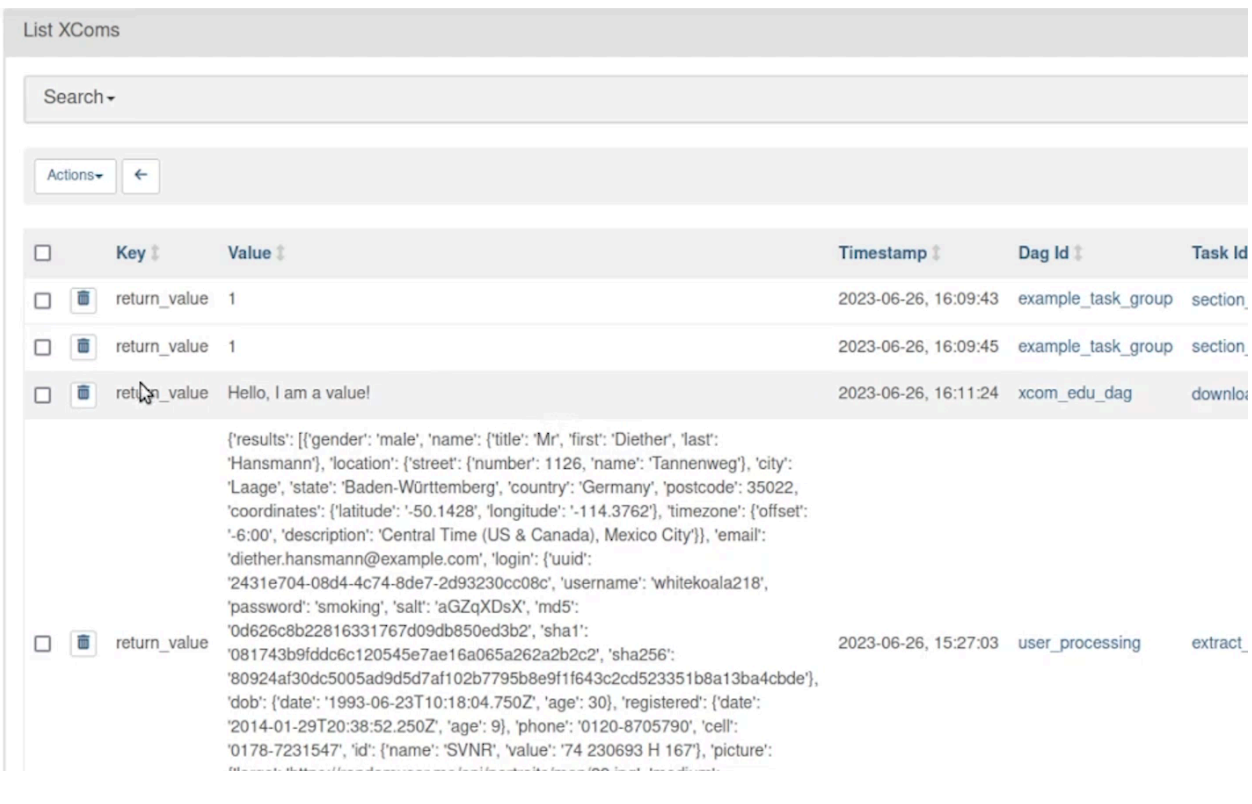
Package Name	Version	Description
apache-airflow-providers-amazon	8.1.0	Amazon integration (including Amazon Web Services (AWS)).
apache-airflow-providers-celery	3.2.0	Celery
apache-airflow-providers-cncf-kubernetes	7.0.0	Kubernetes
apache-airflow-providers-common-sql	1.5.1	Common SQL Provider
apache-airflow-providers-docker	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
	4.4.1	Hypertext Transfer Protocol (HTTP)

https://airflow.apache.org/docs/apache-airflow-providers-celery/3.2.0/

List Pool по умолчанию – 128:

Pool	Slots	Running Slots	Queued Slots	Scheduled Slots
default_pool	128	0	0	0

List XComs. Здесь хранятся XCom’ы:



List XComs

Search

Actions

Key	Value	Timestamp	Dag Id	Task Id
return_value	1	2023-06-26, 16:09:43	example_task_group	section_
return_value	1	2023-06-26, 16:09:45	example_task_group	section_
return_value	Hello, I am a value!	2023-06-26, 16:11:24	xcom_edu_dag	downloa
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': {'uuid': '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 тасков, 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 таска занимают много времени, а где много фэйлов.

Как вам урок? 😠 😐 🔥

Изучил, далее >