

---

---

# Разработка современных витрин данных и ETL- процессов

Астахов Сергей ИУ6-22М

---

# Agenda

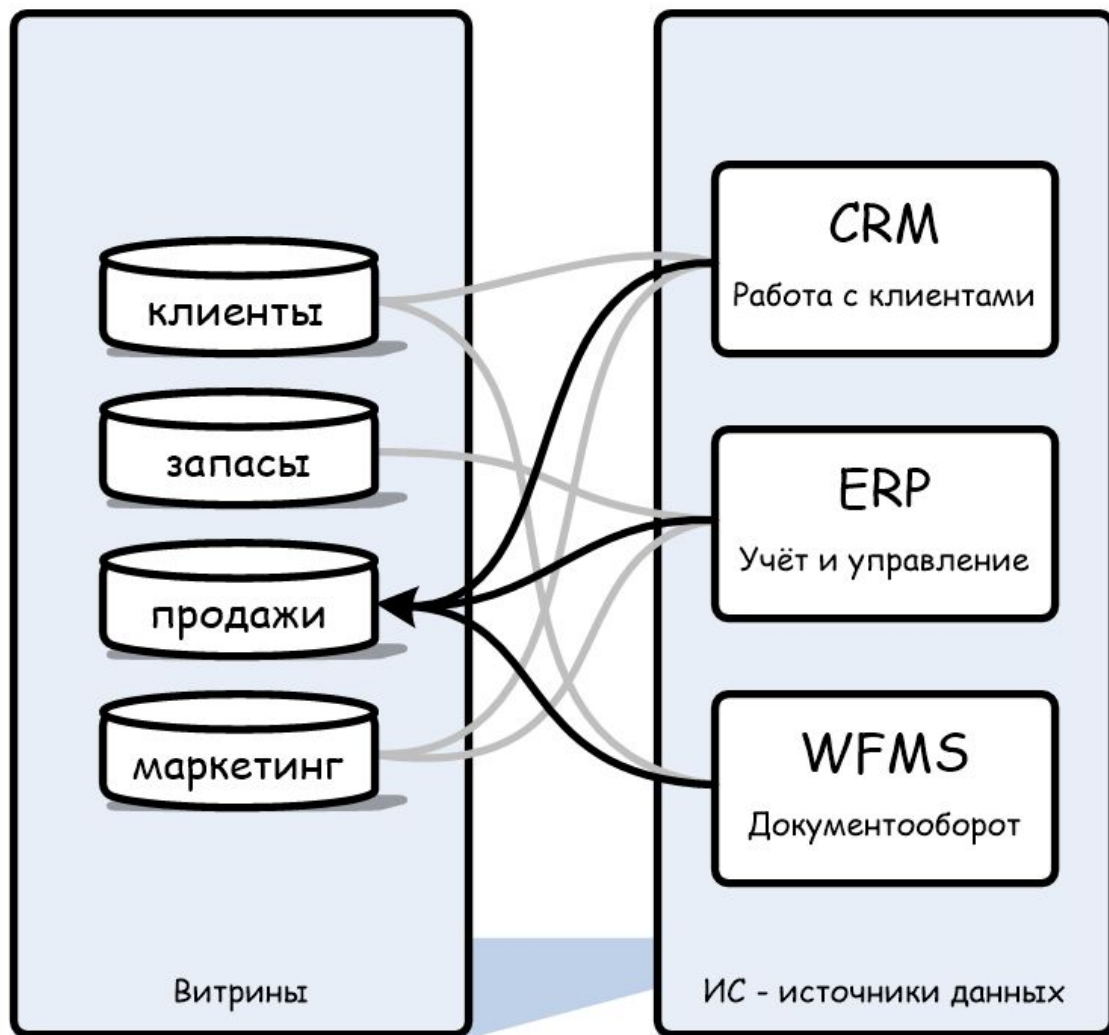
- Введение в витрины данных и ETL-процессы
- Версионность данных (SCD)
- Оркестрация ETL-процессов с Apache Airflow

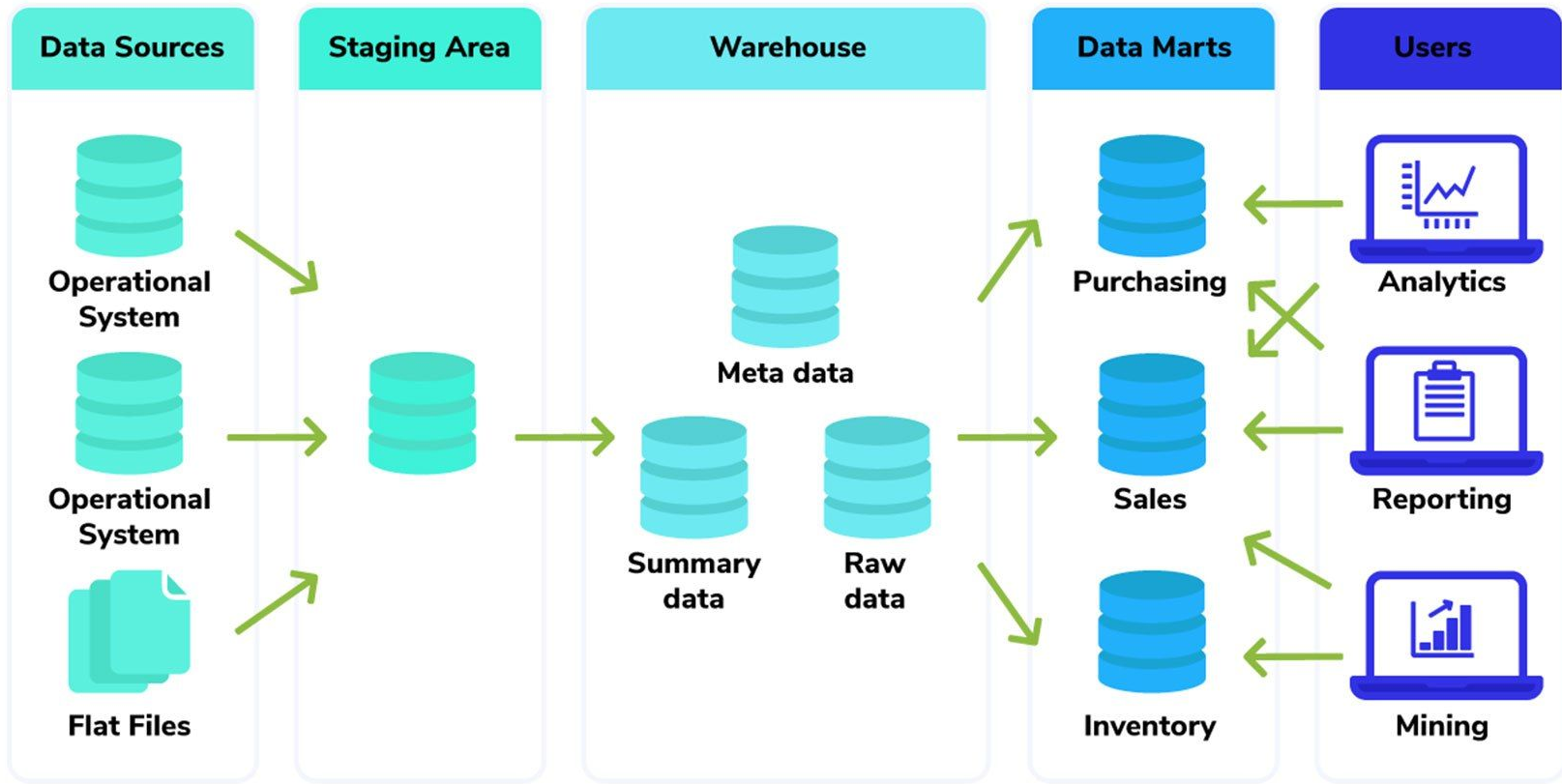
---

---

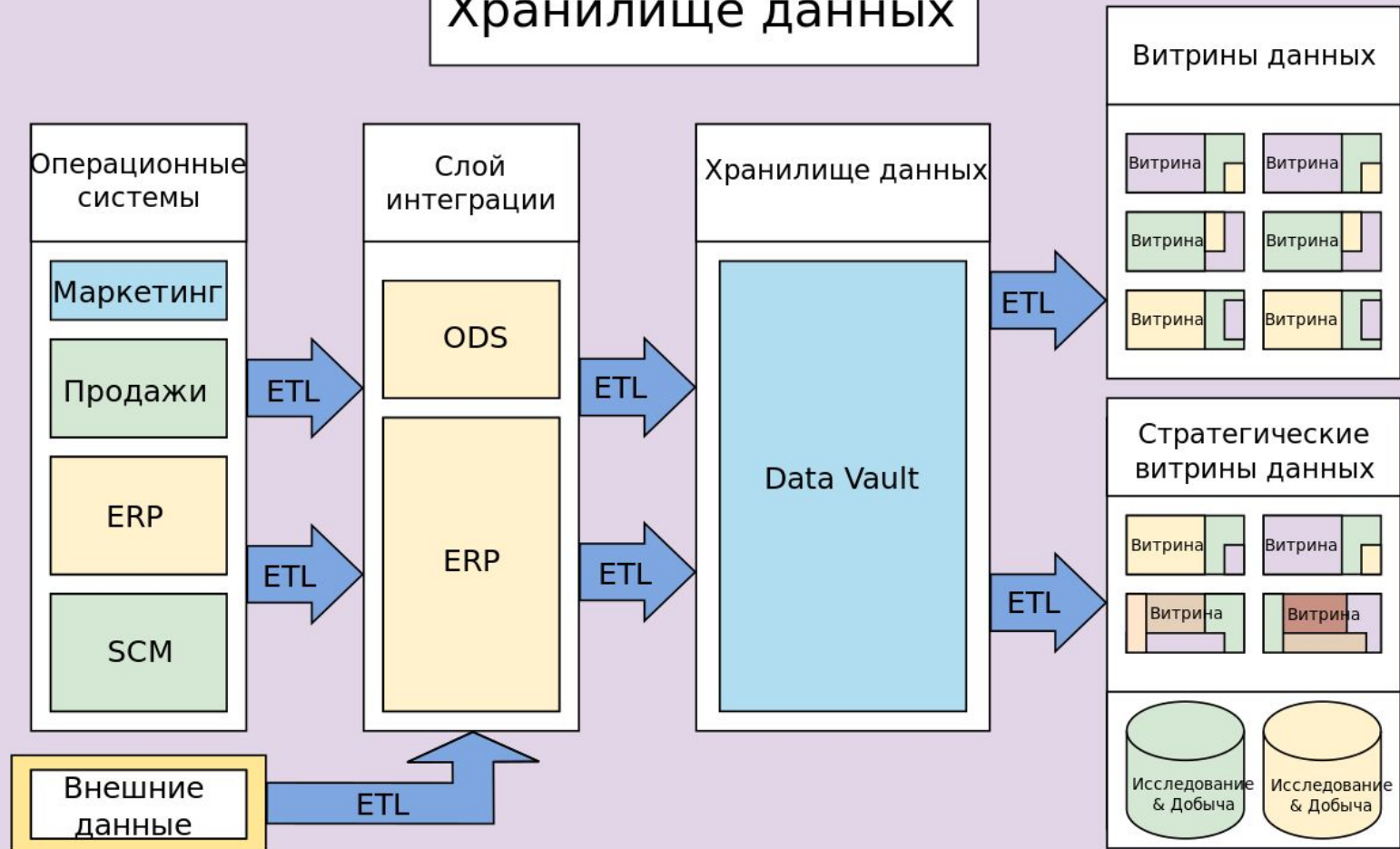
# Введение в витрины данных

---



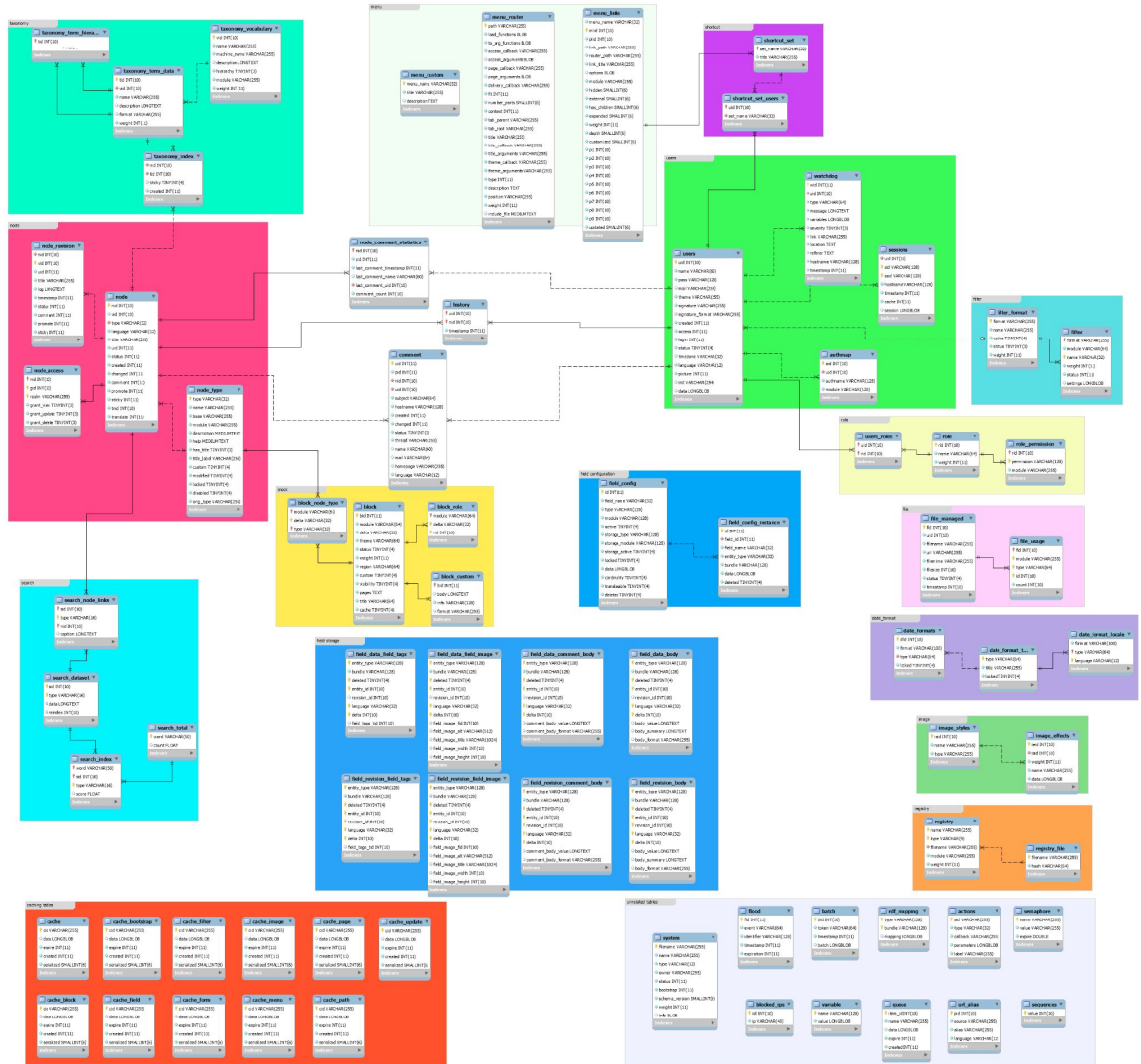


# Хранилище данных



# Что не является витриной данных?

- **View** – передает идею, но слишком сложно проектировать (обычно код витрины – это тысячи строк)
  - **Иерархия view** – вычисляется при каждом обращении
  - **Иерархия materialized view** – вычисляется при каждом изменении, причем полностью
-





---

# Версионность данных (SCD)

---

# SCD-2

ID	NAME	POSITION_ID	DEPT	DATE_START	DATE_END
1	Коля	21	2	11.08.2010 10:42:25	01.01.9999
2	Денис	23	3	11.08.2010 10:42:25	01.01.9999
3	Борис	26	2	11.08.2010 10:42:25	01.01.9999
4	Шелдон	22	3	11.08.2010 10:42:25	01.01.9999
5	Пенни	25	2	11.08.2010 10:42:25	01.01.9999

SCD-0 - данные не меняются

SCD-1 - данные полностью перезаписываются

SCD-2 - т.н. effective-версионность (effective\_from/to)

---

# SCD-3

	ID	UPDATE_TIME	LAST_STATE	CURRENT_STATE
1	1	11.08.2010 12:58:48	0	1
2	2	11.08.2010 12:29:16	1	1

Имеет ограниченную историю

---

---

# SCD-4

ID	NAME	POSITION_ID	DEPT
1	Коля	21	2
2	Денис	23	3
3	Борис	26	2
4	Шелдон	22	3
5	Пенни	25	2

Исторические данные хранятся в отдельной таблице

---

---

# SCD-4

ID	NAME	POSITION_ID	DEPT	DATE
1	Коля	21	1	11.08.2010 14:12:13
2	Денис	23	2	11.08.2010 14:12:13
3	Борис	26	1	11.08.2010 14:12:13
4	Шелдон	22	2	11.08.2010 14:12:13

Исторические данные хранятся в отдельной таблице

---

# SCD-6 (Гибрид)

VERSION	ID	NAME	POSITION_ID	DEPT	DATE_START	DATE_END	CURRENT
1	1	Коля	21	2	11.08.2010 10:42:25	01.01.9999	1
1	2	Денис	23	3	11.08.2010 10:42:25	01.01.9999	1
1	3	Борис	26	2	11.08.2010 10:42:25	11.08.2010 11:42:25	0
2	3	Борис	26	2	11.08.2010 11:42:26	01.01.9999	1

# SCD-6. Пример

user_id	zarplata_amt	eff_from	eff_to	version_id	deleted_flg
Vasya	45.000	2001	2999	1	[]
Vasya	60.000	2005	2999	2	[]
Vasya	null	2007	2999	3	[v]

user_id	zarplata_amt	eff_from	eff_to
Vasya	45.000	2001	2004
Vasya	60.000	2005	2006

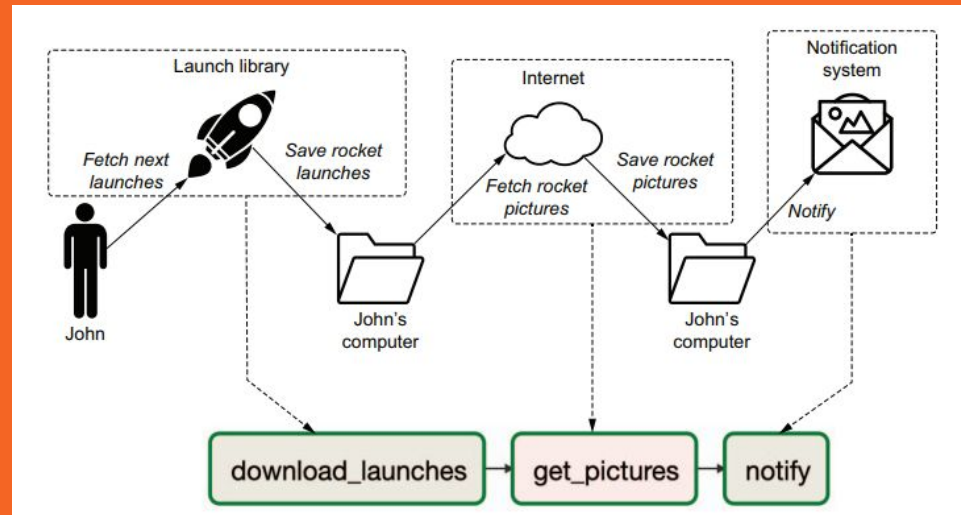
---

# Оркестрация ETL- процессов с Apache Airflow

---



# Пример DAG



DAG – порядок выполнения задач

---

# Пример DAG

with  
DAG('download\_rocket\_launches',schedule\_interval=None,default\_args=default\_args)  
as dag:

```
# шаг 1. Загрузка списка запусков
download_launches=BashOperator(
    task_id='download_launches',
    bash_command="curl -o /tmp/launches.json -L
'https://ll.thespacedevs.com/2.0.0/launch/upcoming/'",
)

# шаг 2. Загрузка фото запусков
get_pictures=PythonOperator(
    task_id='get_pictures',
    python_callable=_get_pictures
)
```

---

---

# Пример DAG

```
# шаг 3. Сообщение в консоль
notify=BashOperator(
    task_id='notify',
    bash_command = 'echo "There are now $(ls /tmp/images/ | wc -l) images."'
)
```

```
# зависимости
download_launches >> get_pictures >> notify
```

---







# UI. Список DAG

## DAGs

All 26 Active 10 Paused 16

Filter DAGs by tag

Search DAGs

 DAG	Owner	Runs 	Schedule	Last Run 	Recent Tasks 	Actions	Links
<input checked="" type="checkbox"/> example_bash_operator example example2	airflow	<div><div>2</div><div></div><div></div></div>	0 0 ***	2020-10-26, 21:08:11 	<div><div>6</div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>	<a href="#">▶</a> <a href="#">↺</a> <a href="#">🗑️</a> ...	
<input checked="" type="checkbox"/> example_branch_dop_operator_v3 example	airflow	<div><div></div><div></div><div></div></div>	* / 1 * * * *		<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>	<a href="#">▶</a> <a href="#">↺</a> <a href="#">🗑️</a> ...	
<input type="checkbox"/> example_branch_operator example example2	airflow	<div><div></div><div>1</div><div></div></div>	@daily	2020-10-23, 14:09:17 	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>	<a href="#">▶</a> <a href="#">↺</a> <a href="#">🗑️</a> ...	

# UI. Исполнение DAG

The screenshot displays the Apache Airflow web interface. The browser address bar shows the URL `localhost:8080/graph?dag_id=download_rocket_launches`. The interface includes a top navigation bar with links for DAGs, Security, Browse, Admin, and Docs. The main header shows the DAG name `download_rocket_launches` with a status of `running` and a schedule of `None`. Below the header, there are tabs for Tree View, Graph View (selected), Task Duration, Task Tries, Landing Times, Gantt, Details, and Code. A filter bar allows selecting a specific run (e.g., `manual__2020-12-17T21:15:02.613390+00:00`) and a layout (e.g., `Left > Right`). A legend at the bottom shows various task statuses: `queued`, `running`, `success`, `failed`, `up_for_retry`, `up_for_reschedule`, `upstream_failed`, `skipped`, `scheduled`, and `no_status`. The DAG graph at the bottom shows three tasks: `download_launches`, `get_pictures`, and `notify`, connected in a linear sequence. The `download_launches` and `get_pictures` tasks are highlighted with green borders, indicating they are currently running.

```
graph LR; download_launches --> get_pictures --> notify
```

# UI. История DAG

The screenshot displays the Apache Airflow web interface for a specific DAG. The browser address bar shows the URL `localhost:8080/tree?dag_id=download_rocket_launches`. The interface includes a navigation bar with links for DAGs, Security, Browse, Admin, and Docs. The main header shows the DAG name `download_rocket_launches` with a description: "Download rocket pictures of recently launched rockets." and a schedule of `@daily`. Below the header, there are tabs for different views: Tree View (selected), Graph View, Task Duration, Task Tries, Landing Times, Gantt, Details, and Code. A filter bar allows selecting a date range (2020-12-16T00:00:00Z) and the number of runs (25). A legend at the bottom identifies task operators (BashOperator, PythonOperator) and their states (queued, running, success, failed, up\_for\_retry, up\_for\_reschedule, upstream\_failed, skipped, scheduled, no\_status). The DAG structure is shown on the left, and the task state over time is shown on the right.

**DAG structure**

```
graph TD; DAG[DAG] --> download_launches[download_launches]; download_launches --> get_pictures[get_pictures]; get_pictures --> notify[notify];
```

**Task state over time**

The task state over time view shows a Gantt chart for the DAG. The chart displays the execution of tasks over time, with columns for each day (Dec 06, Dec 13). The tasks are represented by colored bars: green for success, red for failed, yellow for up\_for\_retry, blue for up\_for\_reschedule, orange for upstream\_failed, pink for skipped, and brown for scheduled. The chart shows that the DAG has been executed multiple times, with most tasks completing successfully.

# UI. Логи DAG

```
*** Reading local file: /opt/airflow/logs/download_rocket_launches/notify/2020-12-17T21:15:02.613390+00:00/1.log
[2020-12-17 21:15:30,917] {taskinstance.py:826} INFO - Dependencies all met for <TaskInstance: download_rocket_launches.notify 2020-12-17T21:15:02.613390+00:00>
[2020-12-17 21:15:30,923] {taskinstance.py:826} INFO - Dependencies all met for <TaskInstance: download_rocket_launches.notify 2020-12-17T21:15:02.613390+00:00>
[2020-12-17 21:15:30,923] {taskinstance.py:1017} INFO -

-----
[2020-12-17 21:15:30,923] {taskinstance.py:1018} INFO - Starting attempt 1 of 1
[2020-12-17 21:15:30,923] {taskinstance.py:1019} INFO -

-----
[2020-12-17 21:15:30,931] {taskinstance.py:1038} INFO - Executing <Task(BashOperator): notify> on 2020-12-17T21:15:02.613390+00:00
[2020-12-17 21:15:30,933] {standard_task_runner.py:51} INFO - Started process 1483 to run task
[2020-12-17 21:15:30,937] {standard_task_runner.py:75} INFO - Running: ['airflow', 'tasks', 'run', 'download_rocket_launches', 'notify', '2020-12-17T21:15:02.613390+00:00']
[2020-12-17 21:15:30,938] {standard_task_runner.py:76} INFO - Job 6: Subtask notify
[2020-12-17 21:15:30,969] {logging_mixin.py:103} INFO - Running <TaskInstance: download_rocket_launches.notify 2020-12-17T21:15:02.613390+00:00>
[2020-12-17 21:15:30,993] {taskinstance.py:1230} INFO - Exporting the following env vars:
AIRFLOW_CTX_DAG_OWNER=airflow
AIRFLOW_CTX_DAG_ID=download_rocket_launches
AIRFLOW_CTX_TASK_ID=notify
AIRFLOW_CTX_EXECUTION_DATE=2020-12-17T21:15:02.613390+00:00
AIRFLOW_CTX_DAG_RUN_ID=manual__2020-12-17T21:15:02.613390+00:00
[2020-12-17 21:15:30,994] {bash.py:135} INFO - Tmp dir root location:
/tmp
[2020-12-17 21:15:30,994] {bash.py:158} INFO - Running command: echo "There are now $(ls /tmp/images/ | wc -l) images."
[2020-12-17 21:15:31,002] {bash.py:169} INFO - Output:
[2020-12-17 21:15:31,006] {bash.py:173} INFO - There are now 2 images.
[2020-12-17 21:15:31,006] {bash.py:177} INFO - Command exited with return code 0
[2020-12-17 21:15:31,021] {taskinstance.py:1135} INFO - Marking task as SUCCESS. dag_id=download_rocket_launches, task_id=notify, execution_date=2020-12-17T21:15:02.613390+00:00
[2020-12-17 21:15:31,037] {taskinstance.py:1195} INFO - 0 downstream tasks scheduled from follow-on schedule check
[2020-12-17 21:15:31,070] {local_task_job.py:118} INFO - Task exited with return code 0
```

---

**Спасибо  
за внимание**

---





**Спасибо  
за внимание**