Backfill, catchup (link)

возможность выполнения за «прошлые" execution_date

catchup = True (default)

Вкл или выкл выполнения пропущенных dag run

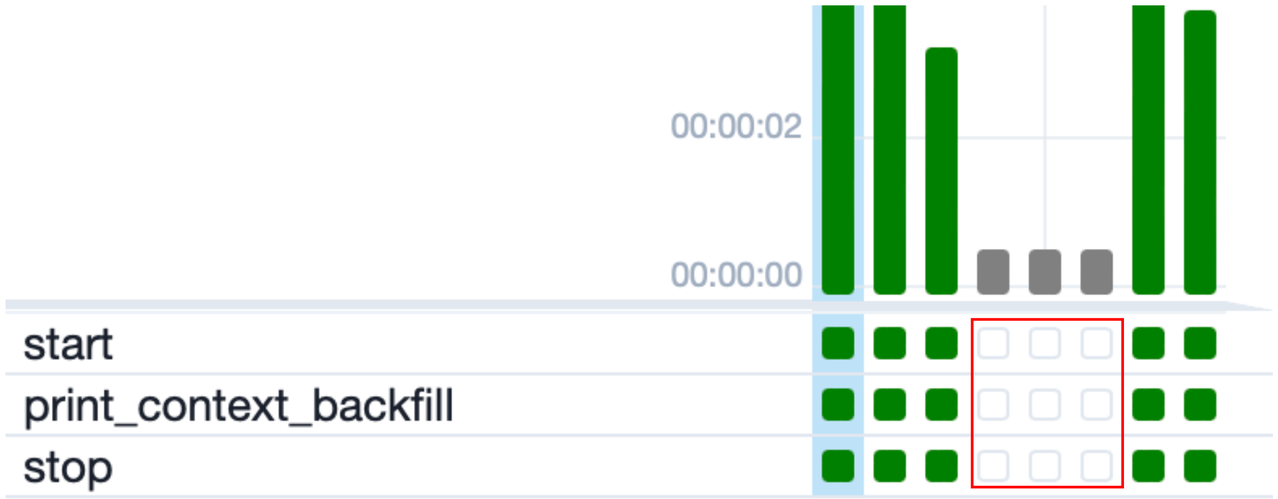
max_active_runs

Максимально допустимое параллельное выполнение задач

start_date/end_date

schedule_interval

Расписание задач



ШКОЛА БОЛЬШИХ ДАННЫХ

Backfill, catchup (link)

возможность выполнения за «прошлые" execution_date

backfill

Используется для выполнения задач за определенный период времени вручную, независимо от настройки catchup

Удобно для случаев, когда необходимо выполнить задачи для конкретного периода, который был пропущен или требует повторного выполнения

catchup

Catchup включен (catchup=True)

Если DAG был отключен или неактивен некоторое время, при его активации или повторной активации Airflow автоматически выполнит все пропущенные задачи.

Удобно для регулярного обновления данных, когда пропущенные интервалы должны быть обработаны

Catchup отключен (catchup=False)

DAG начнет выполнение только с текущего времени, игнорируя все пропущенные интервалы

Удобно для ситуаций, когда пропущенные данные не важны и можно начать выполнение с текущего момента

Backfill

Пример запроса к API для запуска DAG

```
curl -X POST "http://localhost:8080/api/v1/dags/your_dag_id/dagRuns" \
-H "Content-Type: application/json" \
-d '{
  "conf": {},
  "dag_run_id": "manual__backfill_run",
  "execution_date": "2024-06-26T00:00:00+00:00"
}'
```

Пример через CLI

```
airflow dags backfill -s 2024-06-26 -e 2024-06-28 backfill_run,
где -s или --start-date, -e или --end-date
```

Здесь могла бы быть ваша реклама

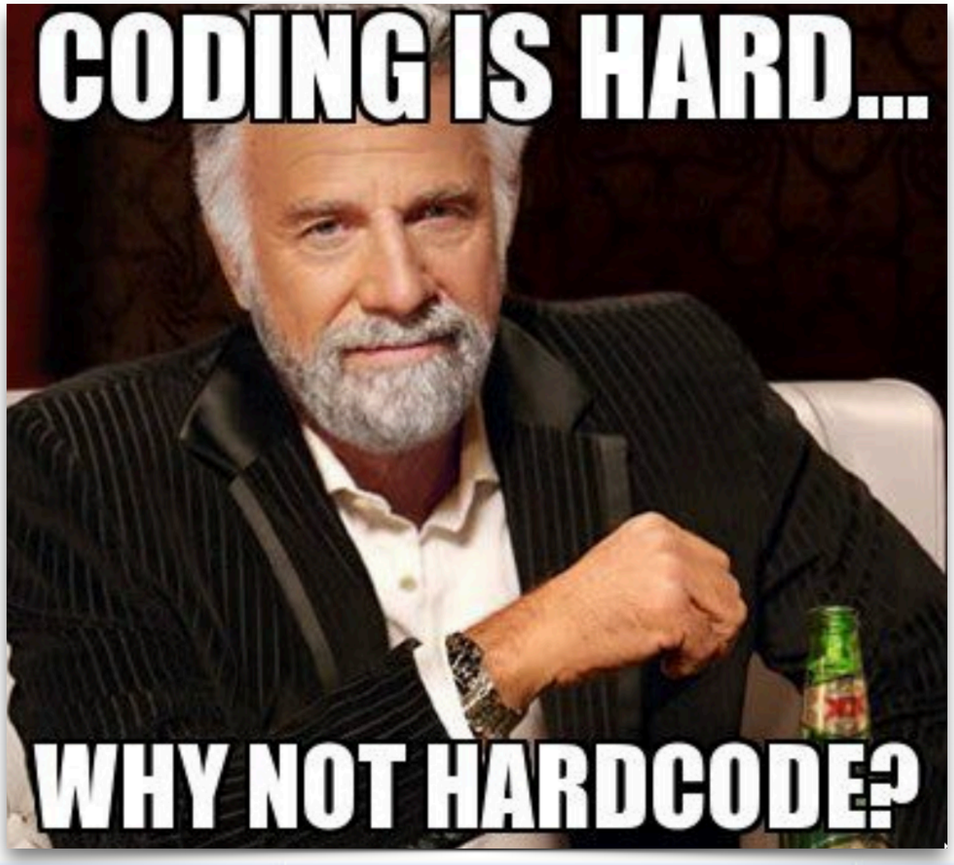
Пример с airflow.models.DagRun

```python
from airflow import settings
from airflow.models import DagRun
from airflow.utils.state import State
from airflow.utils.dates import days_ago
from datetime import datetime, timedelta

# Создаем сессию
session = settings.Session()

dag_id = "first_dag"
start_date = days_ago(10)
end_date = days_ago(1)

# Создаем объект DagRun для каждого дня
current_date = start_date
while current_date <= end_date:
    dag_run = DagRun(
        dag_id=dag_id,
        run_id=f"manual__backfill_run__{current_date.strftime('%Y%m%d')}",
        execution_date=current_date,
        state=State.RUNNING,
    )
    session.add(dag_run)
    current_date += timedelta(days=1)

# Коммитим сессию
session.commit()
```

Variable (link)

Представляет собой глобальное хранилище данных, которое может использоваться в задачах DAG и других компонентах системы.
Позволяет сохранять и извлекать данные, такие как конфигурационные параметры,
сикреты, URL-адреса ресурсов и любую другую информацию, которая может быть необходима в процессе выполнения задач.

Переменные в Airflow могут быть созданы и изменены через веб-интерфейс или с помощью командной строки.
Они могут быть использованы в коде DAG с помощью объекта Variable, который предоставляет методы для работы с переменными.



Variable: без воды (link)

- **пара «key - value»**
- **хранятся в бд airflow**
- **операции (get, delete, set, ...)**

setInWebUI

Admin -> Variables -> +

import

```
from airflow.models import Variable
```

use

```
foo = Variable.get(«foo»)
```

Пример использования:
здесь мы создаем Variable в коде DAG.

```python
from airflow import DAG
from airflow.models import Variable
from datetime import datetime
from airflow.operators.bash import BashOperator


Variable.set("my_variable", "Hello, Airflow!")
my_var = Variable.get("my_variable")

default_args = {
    'owner': 'airflow',
    'start_date': datetime( year: 2021,  month: 1,  day: 1)
}

dag = DAG( dag_id: 'example_dag', default_args=default_args)

task1 = BashOperator(
    task_id='task1',
    bash_command=f'echo {my_var}',
    dag=dag
)

task1
```

???
На каком этапе переменная будет создана?

Variables [«03_practice»]

Текст задания находится на:
/practice_md/03_practice.md

Variables [«04_practice»]

Текст задания находится на:
/practice_md/04_practice.md