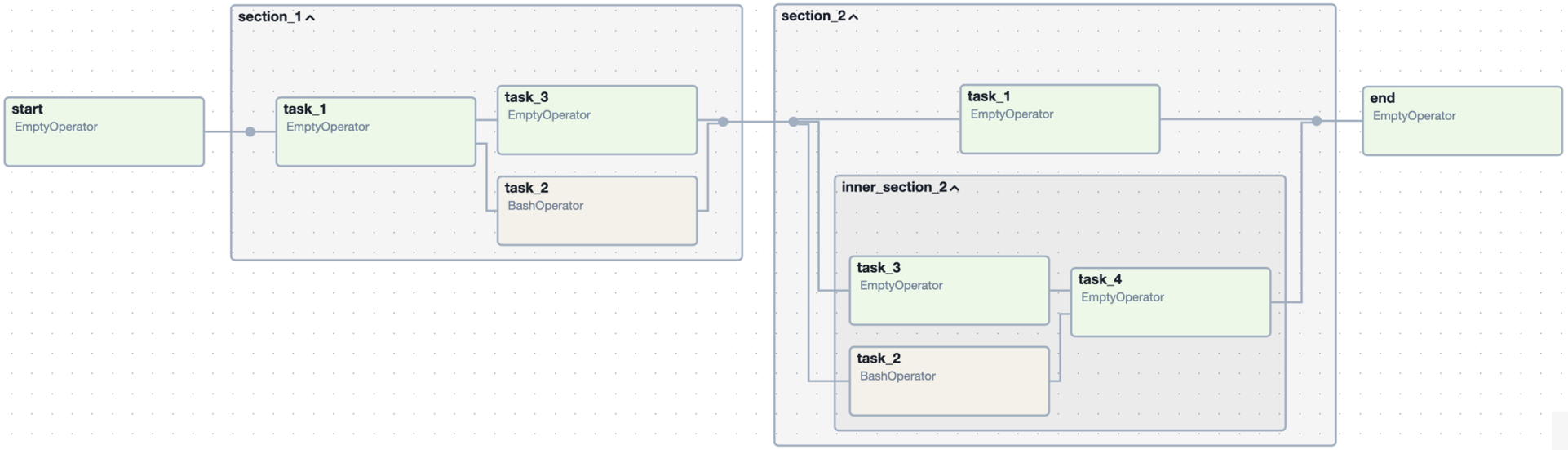
TaskGroup

- Визуальная группировка операторов
- Иерархия групп



Dynamic: XCOM

XCom (cross-communication) позволяет задачам обмениваться небольшими частями данных между собой. Эти данные могут быть строками, числами, списками, словарями и другими типами, которые можно сериализовать в JSON. XComs хранятся в метаданных Airflow и могут использоваться для передачи информации от одной задачи к другой.

- **XCOM - кросс - коммуникация внутри Airflow**
  - ✓ **key, value, task_id ...**
- **push()**
- **pull()**
- **Похоже на Variables**
- **Просмотр через WEB UI**

XCOM vs VARIABLES

XCOM

Локальные значения: XCom представляют собой локальные значения, которые передаются между задачами в пределах одного экземпляра DAG.

Любой тип данных: XCom могут содержать любой тип данных, который можно сериализовать в JSON (строки, числа, списки, словари и т.д.).

VARIABLES

Глобальные значения: Переменные представляют собой глобальные значения, которые можно использовать в любом месте вашего DAG.

Строковые значения: Переменные обычно содержат строковые значения, хотя некоторые значения могут быть сериализованы в другие типы данных.

Управление через UI: Переменные можно создавать, редактировать и удалять через веб-интерфейс Airflow.

Dynamic: Dynamic Task Generation

- **Генерация задач из конфигурации или данных:** Если у вас есть данные или конфигурация, которые определяют список задач или их параметры, динамическое создание позволяет создавать задачи на основе этих данных. Например, если у вас есть список файлов в каталоге, вы можете создать задачи для обработки каждого файла.
- **Управление большим количеством задач:** В некоторых случаях количество задач может быть динамическим и изменяться в зависимости от внешних факторов. Динамическое создание задач позволяет легко масштабировать рабочие процессы и управлять большим количеством задач.
- **Упрощение кода:** Динамическое создание задач позволяет избежать повторения кода и упрощает структуру DAG, особенно когда задачи должны быть созданы на основе данных или конфигурации.

- **Доп. Материал:**
https://airflow.apache.org/docs/apache-airflow/2.3.0/concepts/dynamic-task-mapping.html

Dynamic: Dynamic Task Generation

Разбор моего примера

Datasets in Airflow

Dataset в Apache Airflow представляет собой абстракцию данных, которые могут быть использованы в рамках вашего ETL (Extract, Transform, Load) процесса. Это могут быть файлы, таблицы баз данных, сообщения из очередей, API-ответы и т.д. Airflow предоставляет различные интеграции и операторы для работы с разными источниками данных.

Пример:

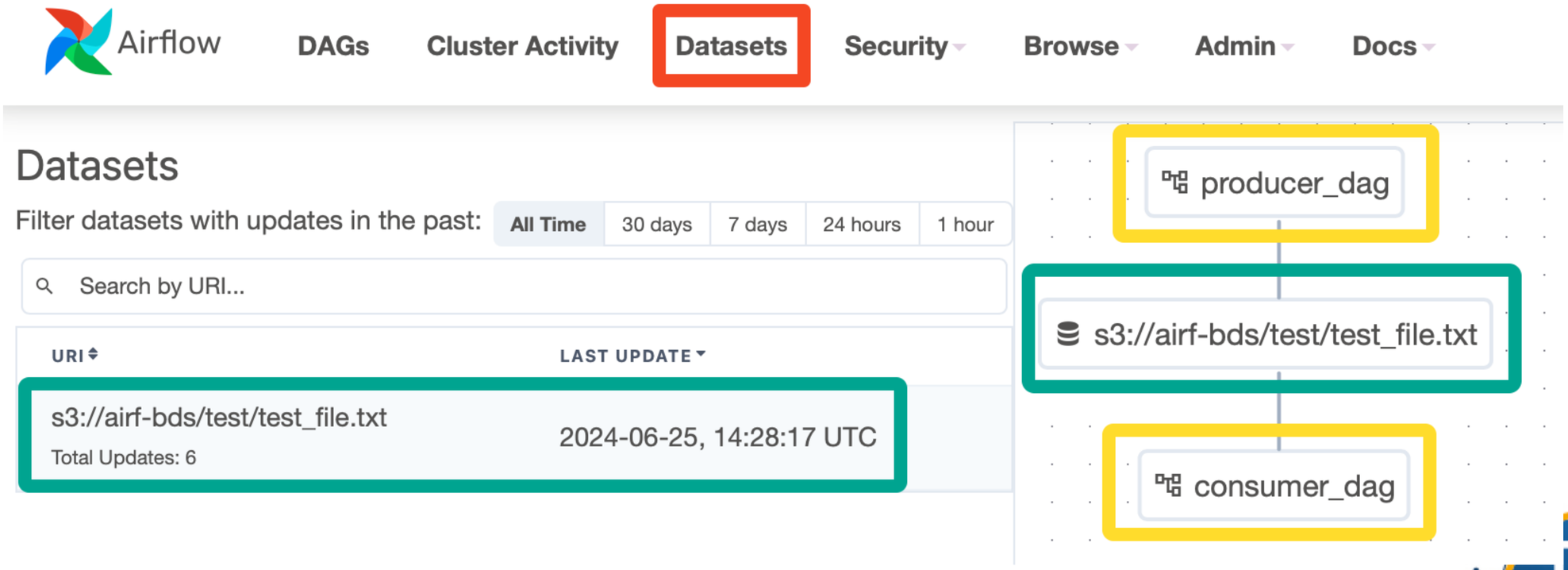
Producer DAG

Этот DAG создает или обновляет файл в хранилище S3 на основе каких-либо данных или событий.

Consumer DAG

Этот DAG читает данные из файла, созданного или обновленного в S3 производителем, и производит запись.

Datasets in Airflow



Datasets in Airflow (link)

Producer DAG

```
create_s3_file_task = PythonOperator(
    task_id='create_or_update_s3_file',
    python_callable=create_or_update_s3_file,
    outlets=[s3_dataset]  # Указываем, что эта задача производит s3_dataset
)
```

Consumer DAG

```
# Создание DAG
with DAG(
        dag_id: 'consumer_dag',
        schedule=[s3_dataset],  # DAG будет запускаться при обновлении s3_dataset
        start_date=datetime( year: 2024,  month: 6,  day: 22),
        catchup=False,
        tags=['lect3', 'dataset']
) as dag:
    read_s3_file_task = PythonOperator(
        task_id='read_from_s3',
        python_callable=read_from_s3,
        inlets=[s3_dataset]  # Указываем, что эта задача потребляет s3_dataset
    )
```

Таким образом, consumer будет запускаться только после того, как producer выполнит свою задачу.

Final Practice

Текст задания находится на:
/practice_md/08_practice.md