airflow 설정중에서 active_dag_runs, max_active_tasks_per_dag 에 대해 설명해줘 (유사한 다른 설정도 있으면 알려주고)
그리고 helm airflow 에서 전역으로 설정하는 예시와 dag에서 지역 설정하는 예도 알려줘

수행자님 기준(쿠버네티스/KubernetesExecutor, Airflow 2.9.2)으로 정리했습니다.

개념 정리 (짧고 정확하게)

- **`max_active_tasks_per_dag`**
  한 **DAG 전체(모든 DAG Run 합산)**에서 동시에 *Running* 될 수 있는 태스크 개수 상한.
  DAG 코드의 **`max_active_tasks`**로 DAG별 오버라이드 가능. 기본값 16. Apache Airflow+1
- **`max_active_runs_per_dag`**
  하나의 DAG가 동시에 가질 수 있는 **활성 DAG Run 개수**의 상한.
  DAG 코드의 **`max_active_runs`**로 DAG별 오버라이드 가능. 기본값 16. Apache Airflow+1
- **자주 같이 만지는 관련 설정**
  - **`[core] parallelism`**: 클러스터(스케줄러 인스턴스당) 전체에서 동시에 실행 가능한 태스크 총량(전 DAG 합산). 기본 32. astronomer.io
  - **Task 단위 `task_concurrency`**: 특정 태스크 ID가 **여러 DAG Run에 걸쳐** 동시에 몇 개까지 실행될지 제한. (오퍼레이터/태스크 인자) Apache Airflow
  - **Pools**: 풀 슬롯으로 동시성 제어(리소스 경합 시 필수). 공식 문서의 동시성 튜닝 문맥에 포함. Apache Airflow
  - **(신규) `max_consecutive_failed_dag_runs_per_dag`**: 연속 실패 N회 시 DAG 자동 pause(2.9+ 실험적, 기본 0=미사용). DAG의 **`max_consecutive_failed_dag_runs`**로 오버라이드 가능. Apache Airflow
  - **스케줄러 튜닝 참고**: **`scheduler.max_tis_per_query`**, **`scheduler.max_dagruns_to_create_per_loop`** 등은 스케줄러의 처리량에 영향. (공식 FAQ/가이드에서 성능 항목으로 안내) Apache Airflow

  직관:
  - “**전세계 슬롯**” = **`parallelism`**
  - “**그 DAG가 쓰는 슬롯**” = **`max_active_tasks_per_dag`** / **`max_active_tasks`**
  - “**그 DAG의 동시 실행 인스턴스 수**” = **`max_active_runs_per_dag`** / **`max_active_runs`**
  - “**같은 태스크ID의 동시 복제수**” = **`task_concurrency`**
  - “**리소스 큐**” = Pools

Helm(공통 전역) 설정 예시

공식 Helm 차트는 **`values.yaml`**의 **`config`** 블록으로 **`airflow.cfg`** 항목을 주입합니다. (섹션 → 키 → 값)
Apache Airflow

```yaml
# values.yaml (발췌)
config:
  core:
    parallelism: 128
    max_active_tasks_per_dag: 32
    max_active_runs_per_dag: 4
    max_consecutive_failed_dag_runs_per_dag: 0   # 원치 않으면 0 유지
  scheduler:
    max_tis_per_query: 1024
    max_dagruns_to_create_per_loop: 20
```

환경변수로도 동일하게 설정 가능:
AIRFLOW__CORE__MAX_ACTIVE_TASKS_PER_DAG, **AIRFLOW__CORE__MAX_ACTIVE_RUNS_PER_DAG** 등.
Apache Airflow

DAG(개별 지역) 설정 예시

전역값을 기본으로 두되, **특정 DAG만 더 느슨/엄격**하게:

```python
from airflow import DAG
from datetime import datetime

with DAG(
    dag_id="etl_heavy_but_safe",
    schedule="@hourly",
    start_date=datetime(2024, 1, 1),
    catchup=False,

    # ↓ 전역 default를 DAG 단위로 오버라이드
    max_active_runs=2,      # 이 DAG는 동시에 2개 Run만
    max_active_tasks=12,    # 이 DAG의 모든 Run 합쳐도 태스크 동시 12개까지만
    dagrun_timeout=None,    # 필요 시 러닝타임 상한 지정 가능
) as dag:
    ...
```

태스크 레벨 동시성(같은 태스크ID 복수 실행 제한)이 필요하면:

```python
from airflow.operators.empty import EmptyOperator

t = EmptyOperator(
    task_id="my_io_bound_task",
    task_concurrency=4,  # 이 태스크ID는 전 DAG Run 합산 4개까지만 동시 실행
)
```

문서상 **max_active_tasks/max_active_runs**는 각각
[core] max_active_tasks_per_dag / **[core] max_active_runs_per_dag**의 DAG 로컬 오버라이드입니다.
Apache Airflow+1

튜닝 팁 (KubernetesExecutor에 딱 맞게)

- **우선순위**: `parallelism` → 풀 슬롯 → `max_active_tasks_per_dag/max_active_runs_per_dag` → `task_concurrency` 순으로 좁혀집니다. 병목을 위에서부터 확인하세요. astronomer.io
- **대량/캐치업 상황**: `max_active_runs_per_dag`를 너무 높이면 잡 폭주로 슬롯을 잠식합니다. 컨트롤러-DAG + Dynamic Task Mapping 구조라면 **`max_active_tasks_per_dag`**를 주 도구로 쓰고, Run 동시성은 1~2로 낮춰도 안정적입니다. (개념은 공식 FAQ/가이드 기준) Apache Airflow
- **실패 폭주 방지(2.9+)**: 실험적이지만 `max_consecutive_failed_dag_runs_per_dag`로 연속 실패 시 자동 pause를 걸어둘 수 있습니다. 생산계엔 0 또는 보수적으로. Apache Airflow

---

필요하시면, **수행자님 클러스터 기준(노드 수/풀 슬롯/현재 DAG 패턴)**으로 적정 숫자(예: `parallelism`, `max_active_tasks_per_dag`)를 역산해서 추천값 뽑아드릴게요.

> dag 실행시에 파라메터를 줄 수 있어?

네, 가능합니다 👍
Airflow에서는 **DAG 실행 시 파라미터(=variables/parameters)**를 넘겨줄 수 있고, 실행된 DAG 안에서 `dag_run.conf`를 통해 읽어옵니다.

---

1. DAG 실행 시 파라미터 전달 방법

CLI

```
bash

airflow dags trigger my_dag \
  --conf '{"date": "2025-09-25", "limit": 100}'
```

REST API

```
bash

curl -X POST "http://<webserver>/api/v1/dags/my_dag/dagRuns" \
  -H "Content-Type: application/json" \
  -u "user:password" \
  -d '{
        "conf": {
          "date": "2025-09-25",
          "limit": 100
        }
      }'
```

Web UI

- DAG → **Trigger DAG w/ config** → JSON 형태 입력

---

2. DAG 코드에서 읽는 방법

```python
from airflow import DAG
from airflow.operators.python import PythonOperator
from datetime import datetime

def print_conf(**context):
    # dag_run.conf를 dict로 접근
    conf = context["dag_run"].conf
    print(">>> DAG Run conf:", conf)

with DAG(
    "example_with_conf",
    start_date=datetime(2024, 1, 1),
    schedule=None,
    catchup=False,
) as dag:

    t1 = PythonOperator(
        task_id="show_conf",
        python_callable=print_conf,
        provide_context=True,
    )
```

실행할 때 **`--conf '{"date":"2025-09-25"}'`** 넣으면, 로그에 **`{'date': '2025-09-25'}`**가 찍힙니다.

---

3. 활용 패턴

- **Dynamic Task Mapping**과 같이 대량의 파라미터를 전달해 태스크 수를 동적으로 조절
- 같은 DAG을 템플릿처럼 쓰되, 입력 값(**`source_table`**, **`target_table`** 등)에 따라 다른 작업 수행
- 운영 환경에서는 API를 붙여서 외부 시스템 이벤트 → DAG 실행 시 파라미터 전달 흐름 구축

---

4. DAG 내부 기본값 처리 (안전장치)

```python
def safe_conf(**context):
    conf = context["dag_run"].conf or {}
    date = conf.get("date", "2025-01-01")  # default 값
    print("Running with date =", date)
```

👉 요약: **CLI, API, WebUI에서 JSON conf를 넘겨서 실행 가능하고, DAG 내부에서는 `dag_run.conf`로 읽는다.**

혹시 원하시면, 제가 **Helm + KubernetesExecutor 환경에서 DAG 실행 파라미터를 전달하는 실제 예시 (kubectl exec로 airflow CLI 호출)**도 적어드릴까요?