airflow 의 SparkSubmitOperator 에 대해서 아주 상세한 설명과 사용법, 예시를 부탁해.

좋습니다. SparkSubmitOperator는 “airflow에서 spark-submit 한 번 제대로 감싸서 던지고, 끝까지 모니터링까지 해주는 친구”라고 보시면 됩니다. 개념부터 설정, 자주 쓰는 파라미터, 실전 예시(로컬/YARN/Kubernetes 순)까지 한 번에 정리해 드릴게요.

1) 무엇을 하는가

- **`spark-submit`** 바이너리를 그대로 래핑해서 Spark 애플리케이션을 제출하고(파이썬 파일, JAR 등), 제출 이후 드라이버 상태를 폴링하며 완료까지 기다립니다. 따라서 Airflow 워커(또는 작업이 실행되는 컨테이너)에 **spark-submit가 PATH에 있어야** 합니다. Apache Airflow
- Airflow 4.x대 Spark 프로바이더부터는 **`spark-home`**을 커넥션에 넣는 방식이 제거되고, **실제 PATH에 있는 spark-submit만 허용**됩니다(허용값: **`spark-submit`**, **`spark2-submit`**, **`spark3-submit`**). Apache Airflow+1

2) 설치·호환성 체크

- 패키지: **`apache-airflow-providers-apache-spark`** (최신 5.3.2 기준)
  - Airflow ≥ **2.10.0**, Python 3.10~3.13, PySpark ≥ **3.5.2** 요구. Kubernetes로 쓸 계획이면 **`[cncf.kubernetes]`** 엑스트라를 함께 설치하세요.
  - 예: **`pip install "apache-airflow-providers-apache-spark[cncf.kubernetes]"`** PyPIApache Airflow

3) Spark 커넥션(`spark_default`) 설정

Airflow UI → **Admin** → **Connections** → *Spark Submit* 타입을 추가합니다. 핵심 필드는 아래와 같습니다. Apache Airflow

- **Host**: **`local`**, **`yarn`**, 또는 URL
  - 예) 로컬: **`local`** / YARN: **`yarn`** / Kubernetes: **`k8s://https://kubernetes.default.svc:443`** (또는 API 서버 URL)
- **YARN Queue**(선택), **Deploy mode**(client/cluster), **Spark binary**(**`spark-submit`**/**`spark2-submit`**/**`spark3-submit`**)
- **Kubernetes namespace**(선택, Spark on K8s 시)
- 환경변수로 등록 예시:

```bash
export AIRFLOW_CONN_SPARK_DEFAULT='spark://mysparkcluster.com:80?deploy-mode=cluster&spark_binary=spark3-submit&namespace=spark-jobs'
```

Apache Airflow

참고: SparkSubmitOperator를 쓰려면 **Spark 커넥션이 선행**되어야 합니다. Apache Airflow

4) 자주 쓰는 파라미터(핵심만)

모두 **Jinja 템플릿** 가능합니다(예: `{{ ds }}`, `{{ var.value.foo }}` 등). 전체 목록은 레퍼런스를 보세요. Apache Airflow

- **`application`**: 제출할 앱(JAR, `.py` 등).
- **`conf`**: `spark.*` 설정 딕셔너리(e.g., `{"spark.executor.memory": "2g"}`)
- **`files`**, **`py_files`**, **`jars`**, **`packages`**, **`exclude_packages`**, **`repositories`**: 의존 파일/패키지
- **`executor_cores`**, **`executor_memory`**, **`driver_memory`**, **`num_executors`**, **`total_executor_cores`**
- **`deploy_mode`**: **`client`** 또는 **`cluster`**
- **`application_args`**: 앱 인자 리스트
- **`env_vars`**: **`spark-submit`** 실행 시 환경변수(K8s, YARN 모두 지원)
- **`yarn_queue`**, **`keytab`**, **`principal`**(Kerberos 사용 시)
- **`spark_binary`**: **`spark-submit/spark2-submit/spark3-submit`** 지정
- **`status_poll_interval`**: 클러스터 모드에서 드라이버 상태 폴링 간격(초)

5) 동작 관점에서의 팁

- **client 모드**: 드라이버가 Airflow 워커 쪽에서 뜹니다. 로그가 보기 쉽지만 워커 리소스와 네트워킹 제약을 받습니다.
- **cluster 모드**: 드라이버가 클러스터 측에서 떠서 독립성 좋고 신뢰성이 높습니다. 대신 드라이버 로그 접근은 클러스터 로그(예: YARN UI, K8s 드라이버 Pod)로 봐야 합니다. Operator는 **`status_poll_interval`**로 드라이버 상태를 계속 확인합니다. Apache Airflow

---

6) 실전 예시

(A) 로컬/스탠드얼론: "빠른 연습용"

```
python


from airflow import DAG
from airflow.utils.dates import days_ago
from airflow.providers.apache.spark.operators.spark_submit import SparkSubmitOperator

with DAG(
    dag_id="spark_local_pi",
    start_date=days_ago(1),
    schedule=None,
    catchup=False,
) as dag:
    submit_pi = SparkSubmitOperator(
        task_id="submit_pi",
        conn_id="spark_default",  # Host=local 로 설정해 두었다고 가정
        application="/opt/spark/examples/src/main/python/pi.py",
        name="pi-local",
        conf={"spark.executor.memory": "1g"},
```

```
        executor_cores=1,
        verbose=True,
    )
```

spark-submit가 Airflow 워커 컨테이너/VM의 PATH에 있어야 합니다. Apache Airflow

(B) YARN 클러스터: "프로덕션 전형"

```
python

submit_yarn = SparkSubmitOperator(
    task_id="submit_yarn",
    conn_id="spark_default",   # Host=yarn, Queue=etl, Deploy mode=cluster 등으로 커넥션 구성
    application="/apps/jobs/my_batch.py",
    name="my-batch",
    deploy_mode="cluster",
    application_args=["--date", "{{ ds }}", "--output", "/warehouse/dt={{ ds }}"],
    conf={
        "spark.sql.shuffle.partitions": "200",
        "spark.yarn.maxAppAttempts": "1",
    },
    num_executors=8,
    executor_cores=4,
    executor_memory="6g",
    driver_memory="2g",
    yarn_queue="etl",
    verbose=True,
)
```

- Kerberos 환경이면 **keytab**, **principal**을 파라미터로 넘기거나 커넥션의 Extra에 둡니다. Apache Airflow

(C) Kubernetes (Spark 3 내장 제출) : "Minikube/온프렘 K8s"

K8s에서 Spark를 직접 제출하려면 **드라이버/익스큐터 이미지를 지정**하고, 네임스페이스/SA를 설정합니다.

```
python

submit_k8s = SparkSubmitOperator(
    task_id="submit_k8s",
    conn_id="spark_default",    # Host=k8s://https://kubernetes.default.svc, Namespace=spark-
jobs 등
    application="local:///opt/spark/examples/src/main/python/pi.py",
    name="spark-pi-k8s",
    deploy_mode="cluster",
    conf={
        "spark.kubernetes.container.image": "apache/spark-py:v3.5.1",
        "spark.kubernetes.namespace": "spark-jobs",
        "spark.kubernetes.authenticate.driver.serviceAccountName": "spark",
        # 필요 시 S3/MinIO, Hive, JDBC 등 추가 conf
    },
    executor_cores=2,
    executor_memory="2g",
    driver_memory="1g",
    spark_binary="spark3-submit",
    verbose=True,
)
```

- 위 방식은 "Spark 자체가 제공하는 K8s 모드"로 **spark-submit**이 K8s API에 Spark 드라이버/익스큐터 파드를 만듭니다.

- 반대로, **Spark Operator(CRD: SparkApplication)**를 쓰고 싶다면 **`SparkKubernetesOperator`**(K8s 프로바이더)를 사용하세요. 두 방식은 다릅니다. Apache Airflow

---

7) 템플릿·동적 파라미터 예

운영 날짜별 파티션 처리처럼 런타임 값을 넣을 때:

```
python

submit_daily = SparkSubmitOperator(
    task_id="daily_job",
    conn_id="spark_default",
    application="/apps/jobs/daily.py",
    application_args=["--date", "{{ ds }}", "--bucket", "{{ var.value.raw_bucket }}"],
    conf={
        "spark.executor.memory": "{{ var.value.exec_mem or '4g' }}",
        "spark.dynamicAllocation.enabled": "true",
    },
    deploy_mode="cluster",
)
```

- **`application`**, **`conf`**, **`files/py_files/jars/packages`**, **`application_args`**, **`env_vars`** 등 주요 필드는 템플릿 대상입니다. Apache Airflow

8) 흔한 이슈와 체크리스트

1. **spark-submit 없음**
   - 에러: **`No such file or directory: 'spark-submit'`** → Airflow 이미지/VM에 Spark를 설치하거나, PATH에 추가하세요. (4.0.0+부터 **`spark-home`**로 우회 불가) Apache Airflow
2. **클러스터 모드 로그 못 봄**
   - YARN: YARN UI에서 드라이버/컨테이너 로그 확인.
   - K8s: 드라이버 Pod 로그(**`kubectl logs`**) 확인. Operator는 상태 폴링만 합니다. Apache Airflow
3. **의존성 누락(드라이버/JDBC 등)**
   - **`jars`** 또는 **`packages`**로 드라이버를 제공하세요(예: MySQL **`mysql-connector-j`** Maven 좌표). Apache Airflow
4. **프로바이더/버전 미스매치**
   - 현재 프로바이더 최신은 5.3.2. Airflow/파이썬/파이스파크 요구사항을 맞추세요. K8s 연동 기능을 모두 쓰려면 **`apache-airflow-providers-cncf-kubernetes`**도 함께. PyPI
5. **K8s 권한**
   - 클러스터 모드에서 드라이버가 리소스를 만들 수 있어야 합니다(ServiceAccount, RoleBinding). (SparkKubernetesOperator를 쓸 경우 CRD 기반 워크플로) Apache Airflow

9) 언제 SparkSubmitOperator를 쓰고, 언제 다른 걸 쓰나

- **SparkSubmitOperator**: 기존 **`spark-submit`** 방식을 Airflow에서 그대로, 단일 잡 제출·모니터링이 목적일 때. (로컬/Standalone/YARN/K8s 모두 가능) Apache Airflow

- **SparkKubernetesOperator**(별도): K8s의 **Spark Operator(CRD)**로 SparkApplication을 생성/관리하고 싶을 때(제거/패치/로그 수집 등 K8s 워크플로와 밀결합). Apache Airflow

빠른 스타터 템플릿

```
python


from airflow import DAG
from airflow.utils.dates import days_ago
from airflow.providers.apache.spark.operators.spark_submit import SparkSubmitOperator

with DAG("spark_starter", start_date=days_ago(1), schedule=None, catchup=False) as dag:
    spark_job = SparkSubmitOperator(
        task_id="spark_job",
        conn_id="spark_default",
        application="/opt/spark/examples/src/main/python/pi.py",
        name="spark-starter",
        deploy_mode="cluster",                  # 필요에 따라 client/cluster
        conf={"spark.executor.memory": "2g"},  # 예시
        verbose=True,
    )
```

사용 전: (1) **`spark_default`** 커넥션 생성 (2) Airflow 워커에서 **`spark-submit`** 호출 가능 확인 (3) 클러스터별 권한·네트워킹 확인. Apache Airflow+1

필요하시면 **Minikube + Airflow(Helm)에서 Spark on K8s**를 바로 돌릴 수 있게, 네임스페이스/SA/이미지까지 포함한 "손에 쥐고 바로 실행" 템플릿도 만들어 드릴게요. 운영 환경에서 무엇을 쓰실지(야른 vs K8s, 드라이버 모드, 이미지/저장소)만 알려주시면 거기에 맞춰 정리해 드리겠습니다.