airflow - DAG 사용자 변경 방법

> 쿠버네티스에서 airflow를 airflow 사용자로 띄웠거든. 그런데 dag를 실행할때 특정 사용자로 실행할 수 있어?

쿠버네티스 위에서 Airflow 전체를 **`airflow`** 유저(컨테이너 안 UID)로 띄운 상태라는 거죠. 그런데 개별 DAG task를 다른 사용자로 실행하고 싶다는 건데... 상황에 따라 몇 가지 길이 있어요.

---

1. KubernetesPodOperator (KPO) 쓸 때

- 가장 깔끔한 방법.
- **`KubernetesPodOperator`** 로 태스크를 실행하면, 그 Pod 안에서 **다른 UID/GID** 를 지정할 수 있습니다.
- **`securityContext`** 를 pod_spec 안에 넣어주면 됩니다.

예:

```
python

from airflow.providers.cncf.kubernetes.operators.kubernetes_pod import KubernetesPodOperator

run_as_other = KubernetesPodOperator(
    task_id="run_as_2001",
    namespace="airflow",
    name="example-2001",
    image="your-custom-image:latest",
    security_context={
        "runAsUser": 2001,
        "runAsGroup": 2001,
        "fsGroup": 2001
    },
    cmds=["/bin/bash", "-c"],
    arguments=["whoami && id && ls -l /mnt/nfs"],
    is_delete_operator_pod=True
)
```

이러면 Airflow 자체는 여전히 **`airflow`** 유저지만, DAG task는 새로운 Pod에서 UID 2001로 돌아갑니다.
→ 보통 NFS 권한 문제 해결할 때 이렇게 많이 써요.

---

2. KubernetesExecutor 쓸 때

- **`KubernetesExecutor`** 도 비슷하게 **`pod_template_file`** 이나 **`pod_override`** 로 **`securityContext`** 넣을 수 있습니다.
- DAG마다, 태스크마다 다르게 줄 수도 있고요.

---

3. Airflow Worker 안에서 sudo/su (비추)

- **`CeleryExecutor/LocalExecutor`** 환경에서라면 worker 컨테이너 안에서 **`sudo -u <user>`** 로 바꿔 실행할 수도 있습니다.
- 근데 쿠버네티스 환경에서 기본적으로는 **`sudo`** 안 깔려 있고, 컨테이너는 보통 rootless라 잘 안 맞습니다. 운영환경에서는 거의 안 쓰는 방식이에요.

정리

Airflow 자체 프로세스는 **`airflow`** 유저로 두고, **DAG 실행시점에만 KPO나 Executor 옵션에서 UID를 바꿔주는 게 정석**입니다.

👉 궁금한 건: 지금 쓰고 계신 실행기는 **`KubernetesExecutor`** 인가요, 아니면 **`KubernetesPodOperator`** 위주로 DAG를 짜고 계세요?
그거에 따라 제가 예시를 좀 더 구체적으로 적어드릴 수 있을 것 같아요.

> 지금 kubernetesExecutor이거든, 그리고 BashOperator 를 1002 uid 로 실행하는 sample을 만들어줘봐

`KubernetesExecutor` 를 쓰면 BashOperator 는 그냥 Airflow worker pod 안에서 돌아서, 컨테이너 기본 유저(**`airflow`**)로만 실행돼요. 그래서 단순히 BashOperator 에 **`uid=1002`** 같은 걸 줄 수는 없습니다.

👉 대신 두 가지 선택지가 있어요:

A. pod_override 로 securityContext 주기 (추천)

`KubernetesExecutor` 는 태스크마다 pod 을 새로 띄우니까, 거기에 override 넣으면 됩니다.

```
python


from airflow import DAG
from airflow.operators.bash import BashOperator
from datetime import datetime
from airflow.kubernetes.pod_generator import PodGenerator

with DAG(
    dag_id="bash_uid_1002_example",
    start_date=datetime(2025, 9, 16),
    schedule_interval=None,
    catchup=False,
) as dag:

    run_as_1002 = BashOperator(
        task_id="run_as_1002",
        bash_command="id && whoami && echo 'Hello from UID 1002'",
        executor_config={
            "pod_override": PodGenerator.construct_pod(
                overrides={
                    "spec": {
                        "securityContext": {
                            "runAsUser": 1002,
                            "runAsGroup": 1002,
                            "fsGroup": 1002,
                        }
                    }
                }
            )
```

```
        },
    )
```

이렇게 하면 해당 태스크용 pod 이 **`uid=1002`** 로 실행돼요.

B. 별도 PodTemplateFile 지정

Airflow Helm 차트 values.yaml 에 pod_template_file 경로를 주고, YAML 안에서 securityContext 를 정의한 뒤 DAG 태스크에 **`executor_config={"pod_override": ...}`** 없이도 적용할 수 있어요.
하지만 DAG 단위로 유저를 바꾸려면 결국 override 방식이 더 유연합니다.

참고

- 위 예제는 BashOperator + KubernetesExecutor 환경에 맞춘 샘플.
- 단, Airflow 2.2 이상이어야 **`pod_override`** 파라미터가 제대로 먹습니다.
- worker pod 이 아니라, executor 가 띄우는 **task pod** 가 UID 1002 로 동작하는 거예요.

혹시 운영환경에서 Helm values.yaml 로 **`pod_template_file`** 을 관리하는 편인가요, 아니면 DAG 코드 쪽에서 바로 **`executor_config`** 쓰는 게 더 편하세요?

> 오류가 PodGenerator.construct pod() got an unexpected keyword argument 'overrides' 이렇게 난다.

맞아요, 그 에러는 Airflow 버전별 API 차이 때문입니다. **`PodGenerator.construct_pod(overrides=...)`** 방식은 지금 쓰시는 버전에서 안 먹어요. **`executor_config`**에 **k8s Python 모델 객체(V1Pod ...)** 를 직접 넣으면 깔끔하게 됩니다.

아래 두 가지 중 하나로 쓰세요. (둘 중 편한 쪽)

방법 1) k8s 모델로 `pod_override` 지정 (추천)

```
python


from datetime import datetime
from airflow import DAG
from airflow.operators.bash import BashOperator

# 핵심: kubernetes 모델 import
from kubernetes.client import (
    V1Pod, V1PodSpec, V1PodSecurityContext, V1Container, V1SecurityContext
)

with DAG(
    dag_id="bash_uid_1002_example",
    start_date=datetime(2025, 9, 16),
    schedule_interval=None,
    catchup=False,
) as dag:

    run_as_1002 = BashOperator(
        task_id="run_as_1002",
```

```
        bash_command="id && whoami && echo 'Hello from UID 1002'",
        executor_config={
            "pod_override": V1Pod(
                spec=V1PodSpec(
                    # Pod 레벨 보안 컨텍스트 (PV/NFS 권한에 영향)
                    security_context=V1PodSecurityContext(
                        run_as_user=1002,
                        run_as_group=1002,
                        fs_group=1002,
                    ),
                    # 컨테이너 레벨도 함께 지정해 두면 확실
                    containers=[
                        V1Container(
                            name="base",  # ⚠ KubernetesExecutor 기본 컨테이너 이름이 보통 "base"
                            security_context=V1SecurityContext(
                                run_as_user=1002,
                                run_as_group=1002,
                            ),
                        )
                    ],
                )
            )
        },
    )
```

- **`name="base"`**를 꼭 맞춰주세요. KubernetesExecutor가 띄우는 태스크 컨테이너 이름이 기본적으로 **`base`**라서 이 이름으로 매칭돼야 override가 적용됩니다.
- **`fsGroup=1002`**는 마운트되는 볼륨(특히 NFS)에 그룹 권한을 맞춰 쓰기 위해 중요합니다.

방법 2) dict로 `pod_override` 지정

버전에 따라 dict merge도 지원됩니다.

```
python


from datetime import datetime
from airflow import DAG
from airflow.operators.bash import BashOperator

with DAG(
    dag_id="bash_uid_1002_example_dict",
    start_date=datetime(2025, 9, 16),
    schedule_interval=None,
    catchup=False,
) as dag:

    run_as_1002 = BashOperator(
        task_id="run_as_1002_dict",
        bash_command="id && whoami && echo 'Hello from UID 1002 (dict)'",
        executor_config={
            "pod_override": {
                "spec": {
                    "securityContext": {
                        "runAsUser": 1002,
                        "runAsGroup": 1002,
                        "fsGroup": 1002
                    },
                    "containers": [
                        {
                            "name": "base",
                            "securityContext": {
                                "runAsUser": 1002,
                                "runAsGroup": 1002
                            }
                        }
                    ]
```

```
                }
            }
        },
    )
```

체크 포인트

- **NFS/PV 권한**: 실제 NFS 디렉토리의 uid/gid(예: 1002/1002)와 맞아야 합니다. 아니면 **`fsGroup`**으로 그룹 쓰기 권한을 열어주는 방향을 고려하세요.
- **PodTemplate 병합**: Helm values에서 **`pod_template_file`**을 쓰고 있다면, 여기 설정과 **병합**되니 충돌되는 필드가 없는지 한번 보세요.
- **PSP/OPA/Policies**: 클러스터 보안 정책이 **`runAsUser`**/**`fsGroup`** 변경을 막고 있진 않은지(에러 이벤트에서 바로 티 납니다).

이렇게 바꾸고 다시 돌려보시면 **`id`** 출력에서 **`uid=1002`**로 나올 거예요. 필요하시면 values.yaml에 넣는 쪽(템플릿 파일)도 예제로 바로 빼드릴게요.