Using Postgres on Airflow for Scalable Data Workflows

Abstract

This document analyzes the choice of database backend for Apache Airflow's metadata database, focusing on PostgreSQL, MySQL/MariaDB, and SQLite. We evaluate each option based on concurrency, availability, feature coverage, and operational risk. The analysis highlights PostgreSQL's strengths in MVCC and lock handling, making it the preferred choice for production deployments with multiple schedulers. MySQL/MariaDB can be viable with recent versions and proper configuration, while SQLite is limited to development use. The document provides practical recommendations for selecting and configuring the metadata database to ensure reliable and efficient Airflow operations.

Glossary

Metadata DB Airflow's operational database storing DAGs, task instances, XComs, jobs, pools, and scheduler state.

MVCC Multi-Version Concurrency Control; readers do not block writers and vice versa.

SKIP LOCKED SQL clause to avoid blocking when a row is locked; used for task claiming.

HA High availability; multiple schedulers and webservers sharing the same metadata DB.

1 Problem and Notation

Goal: choose a database backend $D \in \{\text{PostgreSQL}, \text{MySQL/MariaDB}, \text{SQLite}\}$ to minimize failures and latency for Airflow schedulers. Let:

$$C(D) : \text{concurrency support (rows/sec without deadlock)}$$
$$A(D) : \text{availability under multiple schedulers}$$
$$F(D) : \text{feature coverage of Airflow migrations}$$
$$R(D) : \text{operational risk (lower is better).}$$

Production constraint: Airflow requires an external DB; SQLite is for testing only. :contentReference[oaicite:1]index=1

2 Theoretical Frame and Rival Models

Model 1 (PostgreSQL/MVCC): Strong MVCC, robust `SELECT ... FOR UPDATE SKIP LOCKED`, mature JSONB, transactional DDL. Fits Airflow's locking pattern for task picking.
Model 2 (MySQL/MariaDB/InnoDB): Adequate MVCC; `SKIP LOCKED` arrived late in MariaDB (10.6+). Past deadlocks with multiple schedulers if missing.
Model 3 (SQLite): File-based, single-writer, no HA; intended for local dev.
Note on SQL Server: Backend support removed in Airflow 2.9.0. Not an option for the metadata DB. :contentReference[oaicite:2]index=2

3 Mechanism

Airflow schedulers claim ready tasks using row-level locks:

```
SELECT ... FOR UPDATE SKIP LOCKED
```

This requires backend support for `SKIP LOCKED/NOWAIT` to avoid head-of-line blocking and deadlocks when multiple schedulers run. PostgreSQL provides this reliably; MariaDB needed 10.6+; otherwise HA schedulers are not supported. :contentReference[oaicite:3]index=3

4 Cross-Disciplinary Mapping

DB theory → MVCC and lock semantics. **Distributed systems** → Leaderless workers competing for tasks need non-blocking locks. **Queueing** → `SKIP LOCKED` reduces service time variance by bypassing blocked rows.

5 Thesis–Antithesis–Synthesis

Thesis: PostgreSQL maximizes scheduler throughput and minimizes deadlock risk for Airflow. **Antithesis**: MySQL/MariaDB can match performance if configured on recent versions with correct isolation and SQL features. **Synthesis**: Prefer PostgreSQL by default; choose MySQL/MariaDB only with version/feature checks and load testing. SQLite remains dev-only. :contentReference[oaicite:4]index=4

6 Operational Summary: Pros and Cons by Backend

PostgreSQL

Advantages

- First-class support and community recommendation for Airflow; widely used in managed Airflow. :contentReference[oaicite:5]index=5
- Robust `SKIP LOCKED` and MVCC; stable with multiple schedulers. :contentReference[oaicite:6]index=6
- Strong JSONB, transactional DDL, extensions.

Disadvantages

- Slightly more operational complexity than SQLite.
- Requires tuned VACUUM/autovacuum for heavy XCom churn.

MySQL / MariaDB

Advantages

- Supported for Airflow production backends. :contentReference[oaicite:7]index=7
- Familiar to many ops teams; broad tooling.

Disadvantages

- MariaDB prior to 10.6 lacks `SKIP LOCKED`; multiple schedulers not supported and deadlocks reported. :contentReference[oaicite:8]index=8
- Behavior differences around locking and migrations can require extra tuning; several teams migrated to PostgreSQL after issues. :contentReference[oaicite:9]index=9

SQLite

Advantages

- Zero setup; default for quick testing. :contentReference[oaicite:10]index=10

Disadvantages

- Single-node, single-writer; risk of data loss in production scenarios; no HA. *Use only for development.* :contentReference[oaicite:11]index=11

Microsoft SQL Server

Status: Backend support removed in Airflow 2.9.0; do not choose for the metadata DB. :contentReference[oaicite:12]index=12

7 Operational Steps / Pseudocode

```
if ENV == "dev":
    use SQLite (default)  # quick tryouts only
elif ENV in {"staging", "prod"}:
```

```
    prefer PostgreSQL >= 12
    # ensure:
    # - shared_buffers, work_mem tuned
    # - autovacuum aggressive on xcom tables
    # - connection pooler (pgBouncer) for many webworkers
else:
    if choosing MySQL/MariaDB:
        require MariaDB >= 10.6 or MySQL >= 8.0
        test SELECT ... FOR UPDATE SKIP LOCKED under load
```

Version and feature checks reflect Airflow's HA scheduler needs and documented backend support. :contentReference[oaicite:13]index=13

8 Limits, Uncertainty, Predictions

Limits: Real performance depends on workload mix, XCom volume, and schema migration cadence. **Uncertainty**: Benchmarks vary by hardware and Airflow version; change risk increases across major upgrades. **Prediction**: With multiple schedulers and heavy DAG throughput, PostgreSQL will show fewer lock waits and deadlocks than MariaDB<10.6 and equal or better than MySQL 8.0 under identical tuning. Evidence: Airflow docs on lock semantics and community migrations. :contentReference[oaicite:14]index=14

One-line Reference

"Good architecture is about choosing defaults that fail safely."