

PROJECT LAZARUS

Automated Data Integrity Pipeline



"Trust, but Verify."

Deryn Hurst, Tuntufye Mwakalasya, Maksim
Kharutski, Myckheal Hosang, Nicholas Merjech

Florida International University | CEC | Capstone II | Spring 2026



PROBLEM DEFINITION

The Silent Killer

Checking that a backup file exists is not enough.

Silent data corruption, version skew between tools, and phantom writes can render backups completely useless.

These failures are discovered only at the worst possible moment — during an actual disaster recovery.

72%

of companies with major data loss close within 24 months

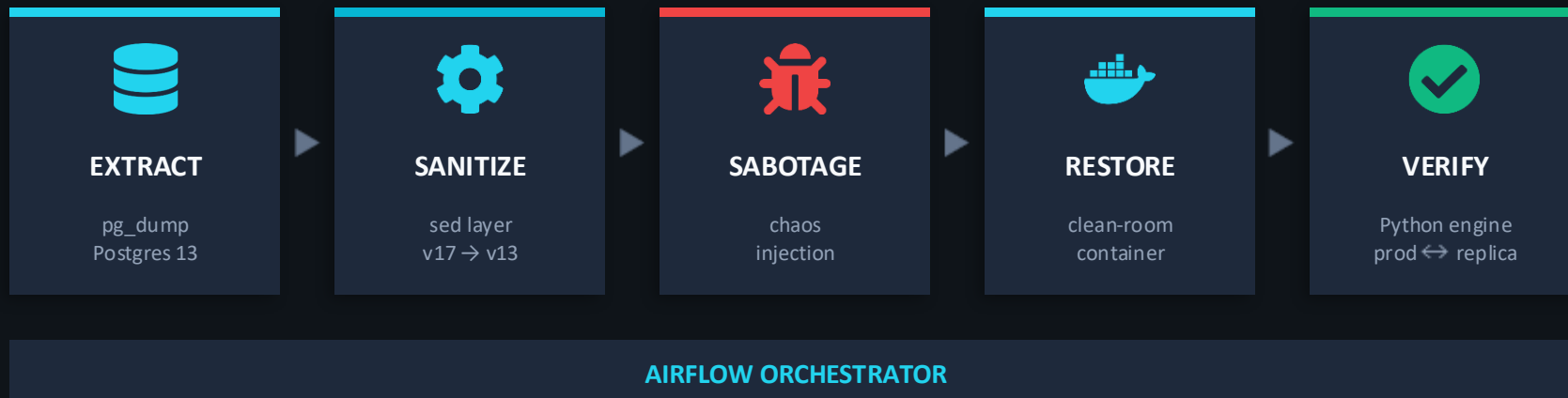
\$4.88M

average cost of a data breach (IBM, 2024)

60%

of backups are incomplete or fail to restore

SYSTEM ARCHITECTURE



The "Data Bridge" Pattern

Airflow workers and database containers run in isolated filesystems. A shared Docker Volume mount on the host acts as a neutral handoff point for backup artifacts between all containerized environments.

</> IMPLEMENTATION HIGHLIGHTS



Handling Version Skew

Airflow ships with Postgres 17 client tools, but production runs v13. Incompatible config parameters cause silent restore failures. A sed stream-editing layer strips incompatible directives from the backup artifact on the fly without touching actual data.



Fail-Fast Reliability

By default, psql ignores errors and continues — a restore can fail while the pipeline stays green. Enforced `ON_ERROR_STOP=1` on all restore commands. If a single byte is wrong, the pipeline crashes immediately.



Secure Secrets

Zero hardcoded credentials. All secrets injected at runtime via Docker Compose from a git-ignored `.env` file using `os.getenv()` pattern.



Tech Stack

Python

Pipeline logic & verification engine

PostgreSQL 13

Production database

Apache Airflow

DAG orchestration

Docker Compose

Container management



DEMONSTRATION OVERVIEW

01

Launch Stack

`docker-compose up -d --build` spins up Postgres, Airflow, and the dashboard containers

02

Trigger DAG

Navigate to Airflow UI at `localhost:8080` and trigger `project_lazarus_verifier` DAG

03

Watch Sabotage

Sabotage task injects rogue rows into the binary backup to simulate real-world corruption

04

Verify Catches It

Verify task connects to both production and replica, runs row-count diff, raises alert on discrepancy

05

Dashboard Results

TypeScript/React dashboard displays verification results, pipeline health, and corruption detection status

TESTING & EVALUATION

Test Scenario	Expected	Result	Status
Clean backup restore	Row counts match	Exact match	✓ PASS
Sabotaged backup (rogue rows)	Pipeline raises alert	Exception raised	✓ PASS
Version skew (v17 → v13)	Sanitize strips params	Restore succeeds	✓ PASS
ON_ERROR_STOP enforcement	Pipeline crashes on error	Immediate failure	✓ PASS
Secrets isolation	No creds in source	grep confirms clean	✓ PASS

6/6

Tests Passed

100%

Corruption Detected

0

False Negatives



CHALLENGES & LESSONS LEARNED

Cross-Container File I/O

Containers have isolated filesystems. Passing backup artifacts between Airflow and Postgres required architecting the Docker volume-mount "Data Bridge" pattern.

Fail Loud, Fail Fast

Silent failures are worse than loud crashes. Every stage should be designed to scream when something is wrong.

Silent Restore Failures

psql defaults to ignoring errors. Discovering that a "successful" restore actually failed required enforcing strict error handling at every stage.

Test Backups, Not Just Files

A backup file existing is meaningless. The only valid test is a full restore into an isolated environment and data comparison.

Postgres Version Mismatch

Airflow's bundled pg_dump (v17) emitted parameters that v13 couldn't parse. Diagnosing this required deep-dive into pg_dump output format differences.

Chaos Engineering Works

Intentionally breaking systems reveals weaknesses that optimistic testing never finds. Adopt the adversarial mindset.



FUTURE WORK



Multi-Database Support

Extend beyond PostgreSQL to MySQL, MongoDB, and other storage engines with pluggable backup strategy interfaces.



Advanced Corruption Modes

Add bit-flip injection, partial truncation, and schema drift simulation to cover more real-world failure scenarios.



Automated Scheduling

CI/CD integration with scheduled nightly verification runs, Slack/PagerDuty alerting on failures, and historical trend tracking.



Schema-Aware Verification

Go beyond row counts: column-level checksums, foreign key integrity validation, and data type drift detection.

PROJECT LAZARUS

If your backup can survive Lazarus, it can survive reality.

Deryn Hurst, Tuntufye Mwakalasya, Maksim Kharutski,
Myckheal Hosang, Nicholas Merjech

github.com/Tmwakalasya/project-lazarus

Florida International University | College of Engineering & Computing

QUESTIONS?