GambillData

Production Databricks Hybrid Architecture Map

Thin notebooks provide orchestration and visibility. Python modules handle reusable business logic. Metadata controls execution. Databricks Asset Bundles standardize deployment.

Core Enterprise Pattern

Layer	Responsibility	What Belongs Here
Thin Orchestrator Notebook	Execution visibility and orchestration	Parameters, control table reads, execution flow, audit logging, task sequencing
Python Utility Engine	Reusable business logic	Transformations, deduplication, merge logic, validation, schema handling, exception handling
Unity Catalog Metadata	Governed execution rules	Control tables, mapping tables, PII rules, schema definitions, load types, lineage metadata
Databricks Asset Bundles	Deployment discipline	Jobs, environments, task configs, CI/CD promotion, version-controlled deployment assets

Recommended Processing Flow

- 1. Control tables determine what should run.
- 2. Mapping metadata defines field-level transformations and governance rules.
- 3. Thin notebooks orchestrate execution and expose operational visibility.
- 4. Python engines execute reusable ingestion, transformation, and quality patterns.
- 5. Databricks Asset Bundles promote the same architecture across environments.

Why This Pattern Holds Up at Scale

- Reduces notebook sprawl and duplicated transformation logic
- Supports metadata-driven onboarding for large table volumes
- Improves debugging visibility during production failures
- Creates cleaner separation between orchestration and business logic
- Improves governance, lineage, and operational consistency
- Makes CI/CD promotion more predictable through DABs

Anti-Patterns

Anti-Pattern	Why It Fails

Monolithic notebooks	Business logic becomes hard to test and maintain
Scripts-only pipelines	Operational visibility becomes harder during incidents
Hardcoded pipeline logic	New tables require repeated engineering effort
Workspace-only deployment	Environment promotion becomes unreliable

Senior Architect Rule

If the logic needs unit tests, it probably does not belong directly in the notebook.

If the logic needs governance, it may belong in metadata.

If the step needs visibility and orchestration, the notebook may be the correct execution surface.

Need Help Designing a Scalable Databricks Architecture?

I work with teams building enterprise-grade data platforms focused on metadata-driven engineering, Medallion architecture modernization, Unity Catalog governance, and deployment strategy.

Schedule a Strategy Session:
https://calendly.com/chris-gambill-gambilldataengineering/strategy-intro

Email:
chris.gambill@gambilldataengineering.com
