ETHIO DATA SOLUTIONS

Databricks Governance Reference Architecture

AWS + Unity Catalog + Terraform | Fictional enterprise pattern

ASSET TYPE	Synthetic reference architecture and control blueprint
STATUS	Review draft | Synthetic scenario | Version 1.0 - August 2026

IMPORTANT DISCLOSURE This demonstration uses a fictional company, invented workloads, and illustrative calculations. It does not represent a customer engagement, customer environment, testimonial, or actual savings result.

Architecture intent

This pattern demonstrates how a fictional organization can provision and operate Databricks on AWS with environment isolation, centralized governance, repeatable Terraform, auditable access, and explicit cost ownership.

Design objective	Pattern
Environment isolation	Separate development, test, and production workspaces; production changes use controlled pipelines
Central governance	One regional Unity Catalog metastore with deliberate workspace bindings and ownership boundaries
Least privilege	Federated identities, groups, service principals, scoped grants, no routine shared credentials
Repeatability	Terraform modules for accounts, workspaces, storage credentials, catalogs, policies, and controls
Operational evidence	Audit delivery, configuration baselines, access reviews, recovery tests, and cost scorecards

Reference boundary

- AWS commercial regions; exact services and features require validation against the client account, Databricks edition, and current documentation.
- The pattern is not a production-ready Terraform repository, security certification, or substitute for threat modeling and client architecture review.
- All names, account numbers, costs, ownership roles, and diagrams are fictional.

1. Logical architecture

IDENTITY & DELIVERY	GOVERNANCE CONTROL PLANE	DATA & COMPUTE
Enterprise IdP SCIM groups MFA / SSO CI/CD service principals	Databricks account Unity Catalog metastore System tables and audit delivery Terraform state and policy modules	Dev / Test / Prod workspaces S3 external locations SQL warehouses and clusters Private connectivity

Identity -> Account controls -> Workspace binding -> Catalog/schema grants -> Auditable data access

AWS account layout

Account	Purpose	Primary controls
Shared security / log archive	Central security evidence and immutable logs	Restricted write paths, retention and monitoring
Data platform non-production	Development and test workspaces, non-production data	Separate IAM roles, budgets, lower-risk experimentation
Data platform production	Production workspace and data plane resources	Change control, restricted roles, stronger monitoring
Terraform state / automation	Remote state, pipelines and deployment identities	State locking, encryption, narrowly scoped assume-role

Workspace pattern

- Use separate workspaces for development, test, and production; avoid using workspaces as the only security boundary.
- Bind catalogs to intended workspaces and assign owners at catalog and schema levels.
- Separate interactive, scheduled-job, and SQL warehouse policies to reflect different risk and cost profiles.

2. Unity Catalog governance model

Layer	Example	Ownership and access
Metastore	us-east-1 enterprise metastore	Platform governance team; no routine analyst ownership
Catalog	prod_sales, prod_finance, nonprod_sandbox	Domain data owner plus platform custodian
Schema	raw, curated, published	Engineering manages raw/curated; business data steward approves published
Object	tables, views, volumes, functions	Least privilege through groups; service principals for workloads
External location	s3://eds-prod-data/domain/...	Storage credential and path scoped to intended catalog/domain

Access model

Principal	Permitted pattern	Avoid
Human users	SSO group membership; time-bound admin elevation	Direct user grants and shared credentials
Job service principals	Per-product or per-pipeline identity with narrow grants	One organization-wide automation identity
Platform automation	Separate account- and workspace-level deployment identities	Using personal tokens in CI/CD
External consumers	Approved sharing mechanism with owner and expiration	Untracked exports and unmanaged copies

Minimum governance controls

- Catalog and schema naming standard tied to environment and data domain.
- Documented owners, grant-review cadence, and separation of owner versus consumer roles.
- Storage credentials and external locations scoped to approved S3 paths and IAM roles.
- Audit evidence retained centrally with alerts for privileged changes and unusual access.

3. Terraform provisioning model

Module boundary	Examples	State / provider scope
AWS foundation	S3, KMS, IAM roles/policies, endpoints, security groups	AWS provider; isolated state per environment
Databricks account	Credential configuration, storage configuration, workspace creation, metastore assignment	Account-level provider and deployment identity
Workspace baseline	Groups, cluster policies, SQL warehouse defaults, secrets integrations	Workspace provider; environment-specific state
Unity Catalog	Catalogs, schemas, grants, storage credentials, external locations	Account/workspace providers as required; explicit dependencies

Pipeline gates

Format, validate, lint, and run policy checks on every change.

Generate a plan using a non-human deployment identity and retain the plan as review evidence.

Require peer approval and environment-specific authorization before apply.

Apply production changes from a protected branch; record actor, commit, plan, and outcome.

Run post-deployment tests for workspace binding, grants, policy attachment, and audit delivery.

State and secret controls

- Encrypt remote state, enable locking/versioning, restrict readers, and avoid secrets in outputs.
- Use workload identity or short-lived credentials; prohibit personal access tokens in repository variables where alternatives exist.
- Separate state to constrain blast radius while preserving deliberate dependencies through outputs or approved data sources.

4. Security, operations, and cost controls

Control area	Baseline control	Evidence
Identity	SSO, SCIM groups, MFA, service-principal inventory	Quarterly access review and group export
Network	Private paths where required; restricted egress; endpoint and DNS validation	Connectivity test and flow/security logs
Data protection	KMS encryption, path-scoped roles, sensitive-data tagging/masking strategy	Configuration baseline and data-owner approval
Compute	Approved runtime versions, cluster policies, restricted instance families	Policy export and exception register
Audit	Account/workspace audit delivery and privileged-change monitoring	Retained logs and alert test
Reliability	Job retry standards, monitoring, recovery procedures, operational ownership	Runbooks and recovery exercise
Cost	Budgets, tags, auto-termination, job clusters, SQL warehouse limits	Monthly scorecard and anomaly dispositions

Illustrative policy decisions

Policy	Default	Exception process
Interactive cluster auto-termination	30 minutes	Time-bound approval with owner and cost impact
Allowed node families	Approved general-purpose list	Performance evidence and architecture review
Production catalog writes	Jobs and approved engineering groups	Break-glass procedure with audit review
SQL warehouse size	Small/medium defaults; autoscaling bounded	Workload test and product-owner approval

5. Implementation roadmap

Phase	Activities	Exit evidence
Discover	Identity, network, AWS accounts, data domains, compliance, current workspaces and costs	Approved requirements and risk register
Foundation	AWS roles/storage/KMS/network, account settings, metastore and workspace assignment	Automated baseline and connectivity evidence
Governance	Catalog/schema design, groups, grants, external locations, audit delivery	Access tests, owner sign-off, audit validation
Operationalize	Cluster/warehouse policies, monitoring, cost controls, runbooks and reviews	Production readiness checklist and operating calendar

Acceptance criteria

- No unmanaged production workspace, catalog, external location, or privileged principal remains outside inventory.
- Representative identities pass positive and negative access tests across environment and domain boundaries.
- Terraform can recreate the approved baseline from protected code and remote state without personal credentials.
- Audit events arrive at the approved destination, are retained, and trigger a tested privileged-change alert.
- Cost owners can explain workspace, compute, and product attribution and have documented anomaly response.

What a real client receives

- Current-state findings and target architecture decision record.
- Terraform module and state design, environment rollout plan, and control matrix.
- Unity Catalog ownership, naming, grants, workspace-binding, and storage-location standards.
- Production-readiness checklist, operational runbooks, and cost-governance scorecard.