**Enterprise Databricks Identity & Access Management**

**1. Document Overview**

**Title:**  
Enterprise Databricks Identity and Access Management (IAM) Architecture and Implementation

**Author:**  
Sri Sivakumar Ramar

**Date:**  
[Date]

**Version:**  
1.0

**2. Objective**

Establish an **enterprise-grade identity and access management framework** for Databricks that supports:

* Centralized user authentication
* Fine-grained access control across workspaces, data, and compute
* Consistent security and compliance alignment
* Operational scalability and ease of administration

**3. Scope**

* Authentication integration with enterprise identity provider (e.g., Azure AD, AWS SSO)
* Role-based access control (RBAC) and attribute-based access control (ABAC)
* Workspace, cluster, and notebook permissions
* Data access controls (Unity Catalog, table-level, column-level, and row-level security)
* Audit logging and monitoring
* Non-functional requirements (security, compliance, scalability)

**4. Architectural Overview**

**4.1 Key IAM Concepts**

|  |  |
| --- | --- |
| **Component** | **Description** |
| **Identity Provider (IdP)** | Enterprise directory service (e.g., Azure Active Directory, AWS IAM Identity Center) |
| **SCIM Provisioning** | Automates user/group provisioning and deprovisioning |
| **SAML / OIDC Federation** | Single Sign-On authentication mechanism |
| **Workspace Permissions** | Controls who can access workspaces and their administrative capabilities |
| **Cluster Policies** | Define allowed configurations for clusters to enforce security and cost controls |
| **Unity Catalog Permissions** | Fine-grained access to catalogs, schemas, tables, and columns |
| **Data Access Controls** | Row-level security, column masking, and data masking policies |
| **Audit Logs** | Centralized logging of authentication events, permission changes, and data access |

**4.2 High-Level Architecture Diagram**

*(You can create a diagram showing the following flows)*

* User Authentication via SSO
* SCIM sync of users and groups
* Workspace Access Policies
* Unity Catalog Data Permissions
* Audit Log Streaming to SIEM

**5. Design Considerations**

**5.1 Authentication Strategy**

* **Single Sign-On (SSO):**
  + Leverage **SAML 2.0** or **OIDC** with Azure AD/AWS SSO.
  + All users authenticate via corporate credentials.
* **Multi-Factor Authentication (MFA):**
  + Enforced at IdP level.

**5.2 User & Group Provisioning**

* **SCIM Integration:**
  + Automate provisioning of users and groups.
  + Deprovision terminated employees automatically.
* **Mapping Groups to Databricks Entitlements:**
  + Admins (Workspace Admins)
  + Developers (Cluster Create/Run)
  + Analysts (SQL Access)
  + Service Accounts (Automation)

**5.3 Workspace Access Control**

| **Level** | **Control** |
| --- | --- |
| **Workspace** | Who can log in and administer |
| **Clusters** | Who can create, attach, run |
| **Jobs** | Who can schedule pipelines |
| **Repos** | Who can manage code repositories |

* **Cluster Policies:**
  + Enforce secure configurations (e.g., credential passthrough, instance types).

**5.4 Data Access Control**

* **Unity Catalog Permissions:**
  + Catalog / Schema / Table / View
* **Row-Level Security:**
  + Example:

sql

CopyEdit

CREATE ROW FILTER rls\_sales AS (region = current\_user\_region());

* **Column Masking:**
  + Example:

sql

CopyEdit

CREATE MASKING POLICY ssn\_mask AS (val STRING) RETURNS STRING ->

CASE WHEN is\_account\_group\_member('compliance') THEN val ELSE '\*\*\*-\*\*-\*\*\*\*' END;

**5.5 Service Principals and Automation**

* Use **Service Principals** for non-human access (e.g., CI/CD deployments).
* Apply **least privilege** permissions.

**5.6 Audit Logging & Monitoring**

* Enable **Audit Log Delivery**:
  + Authentication events
  + Permission grants/changes
  + Data access logs
* Stream logs to **SIEM** (Splunk, Sentinel).

**5.7 Environment Isolation**

* Separate Workspaces for:
  + Development
  + QA
  + Production
* Isolate IAM entitlements per environment.

**6. Non-Functional Requirements**

| **NFR Area** | **Description** |
| --- | --- |
| **Security** | - SSO with MFA - Least privilege RBAC - Encryption in transit and at rest - Strict cluster policies |
| **Scalability** | - Support 5,000+ users - Automated provisioning - Attribute-based access for dynamic control |
| **Availability** | - SSO failover readiness - High-availability IdP integration |
| **Compliance** | - GDPR, HIPAA, SOC2 readiness - Long-term audit log retention (1+ years) |
| **Performance** | - Access permission evaluation <500ms - Provisioning sync latency <15 min |
| **Observability** | - Near-real-time audit log delivery - Alerts on failed logins and permission changes |
| **Cost** | - Cost center tagging by group - Resource usage monitoring to prevent excessive consumption |

**7. Implementation Steps**

**7.1 Prerequisites**

* Enterprise IdP configured (Azure AD, Okta, AWS SSO)
* Service Principal with SCIM API permissions
* Admin account in Databricks

**7.2 SSO Configuration**

1. Generate SAML metadata in Databricks.
2. Create enterprise app in IdP.
3. Configure SAML or OIDC with:
   * Entity ID
   * Reply URL
   * Certificate
4. Test SSO authentication.
5. Enable MFA enforcement in IdP.

**7.3 SCIM Provisioning**

1. Enable SCIM token in Databricks.
2. Configure SCIM endpoint in IdP.
3. Map IdP groups to Databricks entitlements.
4. Validate user sync:
   * User created in IdP
   * SCIM sync to Databricks
   * User can log in via SSO

**7.4 Workspace Permissions**

* Assign Workspace Admins
* Define Cluster Policies
* Grant permissions per role

**7.5 Unity Catalog Permissions**

1. Create catalogs and schemas.
2. Assign groups and grants:

sql

CopyEdit

GRANT USAGE ON CATALOG finance TO `finance\_analysts`

GRANT SELECT ON SCHEMA finance.curated TO `finance\_analysts`

1. Implement masking and row-level policies.

**7.6 Service Principals**

* Create Service Principal tokens for automation.
* Restrict access to specific catalogs and clusters.

**7.7 Audit Log Configuration**

* Configure audit log delivery to secure storage.
* Integrate with SIEM.
* Validate event ingestion.

**7.8 Testing & Validation**

* Test user login flows.
* Validate role-based permissions.
* Confirm audit logs capture events.
* Perform negative tests (unauthorized access attempts).

**7.9 Operational Handover**

* Provide IAM playbooks:
  + Adding/removing users
  + Managing cluster policies
  + Responding to access incidents
* Document escalation paths.

**8. Risks & Mitigations**

| **Risk** | **Mitigation** |
| --- | --- |
| Inconsistent provisioning | Automate SCIM and monitor sync failures |
| Excessive permissions granted | Implement strict approval workflows |
| Orphaned accounts | Schedule regular entitlement reviews |
| SSO misconfigurations | Validate failover IdP configuration |
| Audit log gaps | Continuously monitor delivery health |

**9. Future Enhancements**

* Attribute-based access control (ABAC) policies
* Automated policy as code (Terraform)
* Enhanced lineage integration for data stewardship
* Integration with Data Loss Prevention (DLP) systems

**10. Appendix**

* SAML configuration guides
* SCIM API reference
* Unity Catalog security reference
* Sample Terraform modules

**11. Approval & Sign-Off**

| **Name** | **Role** | **Signature** |
| --- | --- | --- |
| [Name] | Enterprise Data Architect |  |
| [Name] | Security & Compliance Lead |  |
| [Name] | Platform Owner |  |

**Final Architect Notes**

This IAM architecture is designed to:

* **Centralize identity management**
* Enforce **least privilege and compliance**
* Provide **scalable and auditable security controls**
* Support long-term evolution into **attribute-based access** and **policy as code**