# **Technical Design Document (TDD)**

## Structuring Databricks for Success: Folders, Access, Governance

Contents

[Technical Design Document (TDD) 1](#_Toc202356175)

[Title: Organizational Folder and Artifact Structure in Databricks 1](#_Toc202356176)

[1. Purpose 1](#_Toc202356177)

[2. Scope 1](#_Toc202356178)

[3. Assumptions 1](#_Toc202356179)

[4. Folder Structure Design 1](#_Toc202356180)

[5. Naming Conventions 2](#_Toc202356181)

[6. Governance with Unity Catalog 2](#_Toc202356182)

[7. Git Integration & CI/CD 4](#_Toc202356183)

[8. Job Management 5](#_Toc202356184)

[9. Delta Live Tables (DLT) 5](#_Toc202356185)

[10. Permissions and Access Control 6](#_Toc202356186)

[11. Cleanup & Retention 7](#_Toc202356187)

[12. Documentation Standards 8](#_Toc202356188)

[13. Tools & Integrations 9](#_Toc202356189)

[14. Summary 9](#_Toc202356190)

### 1. Purpose

This document outlines the best practices for organizing folders and artifacts in Databricks to ensure clarity, scalability, collaboration, and governance across teams and environments. It is tailored for the implementation of a project: **SBK** **Customer360 Platform**.

### 2. Scope

Applies to all Databricks workspaces used across data engineering, analytics, and ML teams, supporting structured workflows in development, testing, and production environments. This structure supports SBK Customer360 Platform which consolidates sales, support, marketing, and digital footprint data into a unified customer profile.

### 3. Assumptions

* Unity Catalog is enabled for governance.
* Git integration is in place.
* CI/CD workflows are adopted for deployments.
* Customer360 uses Azure Databricks with enterprise-level workspace governance.

### 4. Folder Structure Design

#### 4.1 Top-Level Structure

/Shared  
 └── Customer360  
 ├── dev/  
 ├── test/  
 └── prod/

|  |  |
| --- | --- |
| Dev/ | Development workspace for experimentation, unit testing, and sandbox |
| Test/ | Environment for integration testing and QA validation workflows |
| Prod/ | Production-ready notebooks, jobs, and pipelines for live execution or Alternative (Team-Based) |

/Shared  
 ├── customer360-data-engineering/ **#Contains ingestion scripts, ETL notebooks, DLT pipeline**   
 ├── customer360-analytics/ **# Dashboards, reporting queries, view**   
 └── customer360-mlops/ **# ML models, training workflow,** **Mlflow,** **deployment**

#### 4.2 Subfolders (per Environment)

/Shared/Customer360/dev/  
 ├── notebooks/ **# Contains exploration, transformation, or EDA notebooks**  
 ├── jobs/ # **Databricks job definitions and scheduled tasks**  
 ├── pipelines/ # **Delta Live Tables or structured ETL pipelines** ├── dbt\_models/ # **DBT models and configs if using dbt-core**  
 ├── tests/ # **Data quality and unit/integration tests**  
 └── docs/ # **Markdown files, architecture diagrams, README**

### 5. Naming Conventions

|  |  |
| --- | --- |
| Artifact Type | Naming Convention Example |
| Notebook | 01\_ingest\_crm\_data, 02\_transform\_support\_tickets |
| Job | load\_web\_events\_dev |
| Pipeline | curate\_customer\_gold\_profile\_dlt |
| Table | customer\_bronze, customer\_silver, customer\_gold |
| Test Script | test\_missing\_values\_crm\_customers |

### 6. Governance with Unity Catalog

* Catalogs: sbk\_customer360
* Schemas: bronze, silver, gold
* Access Control: Role-based via Unity Catalog (e.g., marketing\_read, eng\_write)
* Sensitive Data: Apply row/column-level access policies for fields like email, phone, ssn

|  |
| --- |
| Governance with Unity Catalog |
| 1. Create Catalog and Schemas  sql  CopyEdit  -- Create catalog  CREATE CATALOG IF NOT EXISTS sbk\_customer360  COMMENT 'Customer 360 catalog for all customer-related data (ACME brand)';  -- Set owner  ALTER CATALOG sbk\_customer360 OWNER TO `accountadmin`;  -- Create schemas  CREATE SCHEMA IF NOT EXISTS sbk\_customer360.bronze COMMENT 'Raw landing data';  CREATE SCHEMA IF NOT EXISTS sbk\_customer360.silver COMMENT 'Cleaned and transformed data';  CREATE SCHEMA IF NOT EXISTS sbk\_customer360.gold COMMENT 'Curated data for analytics';  2. Grant Role-Based Access Control (RBAC)  sql  CopyEdit  -- Grant read-only access to marketing  GRANT USAGE ON CATALOG sbk\_customer360 TO `marketing\_read`;  GRANT SELECT ON ALL TABLES IN SCHEMA sbk\_customer360.gold TO `marketing\_read`;  -- Grant full access to engineers  GRANT USAGE ON CATALOG sbk\_customer360 TO `eng\_write`;  GRANT CREATE, SELECT, MODIFY ON SCHEMA sbk\_customer360.silver TO `eng\_write`;  -- Optional: Auto-grant on future objects  GRANT SELECT ON FUTURE TABLES IN SCHEMA sbk\_customer360.gold TO `marketing\_read`;  3. Define and Apply Row Access Policy  Goal: Restrict rows by region or department  sql  CopyEdit  -- Create Row Access Policy  CREATE OR REPLACE ROW ACCESS POLICY region\_policy  AS (region STRING)  USING (CURRENT\_USER() IN ('user1@example.com', 'user2@example.com') OR region = 'EMEA');  Apply to table  sql  CopyEdit  ALTER TABLE sbk\_customer360.gold.customer\_profile  ADD ROW ACCESS POLICY region\_policy ON (region);  4. Define and Apply Column Masking for Sensitive Fields  Masking Policy for Email  sql  CopyEdit  CREATE OR REPLACE MASKING POLICY email\_mask  AS (email STRING)  USING (CASE  WHEN is\_account\_group\_member('pii\_access') THEN email  ELSE '\*\*\*MASKED\*\*\*'  END);  Masking Policy for SSN  sql  CopyEdit  CREATE OR REPLACE MASKING POLICY ssn\_mask  AS (ssn STRING)  USING (CASE  WHEN is\_account\_group\_member('pii\_access') THEN ssn  ELSE NULL  END);  Apply to table columns  sql  CopyEdit  ALTER TABLE sbk\_customer360.gold.customer\_profile  ALTER COLUMN email  SET MASKING POLICY email\_mask;  ALTER TABLE sbk\_customer360.gold.customer\_profile  ALTER COLUMN ssn  SET MASKING POLICY ssn\_mask;  Optional: Create Access Groups  In the admin console (or SCIM APIs), create the following access groups:   * marketing\_read * eng\_write * pii\_access   Then assign users to these groups accordingly. |

### 7. Git Integration & CI/CD

**Repo Structure**:

/customer360-repo  
 ├── src/  
 │ ├── notebooks/  
 │ ├── workflows/  
 │ └── utils/  
 ├── tests/  
 ├── docs/  
 └── .github/  
 └── workflows/

Versioning Format: .py for production workflows, .ipynb for exploration.

**/customer360-repo Folder Structure**

|  |  |
| --- | --- |
| Folder Path | Purpose |
| src/ | Source code directory for all core data workflows and assets |
| └── notebooks/ | Reusable notebooks for ingestion, transformation, and exploration |
| └── workflows/ | Production-ready pipeline scripts (DLT, batch jobs, streaming) |
| └── utils/ | Shared utility functions and helper modules (e.g., logging, schema defs) |
| tests/ | Unit tests and data validation scripts for CI pipelines |
| docs/ | Architecture diagrams, onboarding notes, and markdown documentation |
| .github/ | GitHub-specific folder for CI/CD automation |
| └── workflows/ | YAML pipelines for Databricks job deployment, DBT builds, code checks |

### 8. Job Management

* Use Tags: env: dev, project: customer360
* Naming: load\_customer\_demographics\_prod, sync\_support\_data\_test
* Folder Grouping: Jobs organized under /jobs/ per environment with alert policies configured

**Job Management Standards**

|  |  |
| --- | --- |
| Practice | Description |
| Use Tags | Assign tags such as env: dev, env: prod, and project: customer360 to all jobs for better filtering, traceability, and automation triggers. |
| Naming | Use descriptive, lowercase names with underscores for clarity. Example job names: load\_customer\_demographics\_prod, sync\_support\_data\_test. |
| Folder Grouping | Organize jobs under environment-specific folders (e.g., /jobs/dev/, /jobs/prod/) and configure alert policies for monitoring SLA violations or failures. |

This helps ensure:

* Easier job search and filtering in the UI
* Cleaner DevOps workflows with consistent identifiers
* Better observability via alerts and notifications

### 9. Delta Live Tables (DLT)

/Shared/Customer360/prod/pipelines/  
 ├── dlt\_ingest\_web\_events.py  
 ├── dlt\_curate\_crm.py  
 └── dlt\_build\_customer\_gold.py

Use @dlt.table, @dlt.expect decorators for schema enforcement and SLAs.

**Delta Live Tables (DLT) Pipeline Structure**

|  |  |
| --- | --- |
| File Path | Purpose |
| dlt\_ingest\_web\_events.py | Ingests raw website event data from tracking systems into the bronze layer using @dlt.table decorators. |
| dlt\_curate\_crm.py | Applies transformation logic to CRM records, filters invalid rows, and writes curated data to the silver layer. |
| dlt\_build\_customer\_gold.py | Joins CRM, support, and web data to produce a unified customer profile in the gold layer for analytics. |

This layout enables:

* Modular pipeline development by domain/stage
* Easier monitoring and SLA enforcement
* Reusability across environments via parameterized configs

### 10. Permissions and Access Control

* Folder-level and notebook-level ACLs per team
* Unity Catalog governance via group policies
* RBAC for roles like marketing\_analyst, data\_engineer, auditor

**Permissions and Access Control**

|  |  |
| --- | --- |
| Control Mechanism | Description |
| Folder & Notebook ACLs | Set **folder-level and notebook-level access controls** to restrict visibility and edit rights by team or individual contributor. |
| Unity Catalog Group Policies | Leverage **Unity Catalog governance** by applying group-based permissions for catalogs, schemas, and tables (e.g., marketing\_read, eng\_write). |
| Role-Based Access Control (RBAC) | Assign predefined roles like marketing\_analyst, data\_engineer, and auditor to enforce fine-grained access across data layers and tools. |

This ensures:

* Strong data security across dev, test, and prod environments
* Seamless collaboration without permission conflicts
* Auditability aligned with enterprise governance standards

### 11. Cleanup & Retention

* Delta table retention: 30 days for bronze, 90 days for silver
* Notebooks archived to Git monthly
* Job results and logs auto-expired after 14 days using workspace settings

|  |
| --- |
| **Clean UP** |
| **1. Delta Table Retention Policy**  Set **Time Travel retention** using ALTER TABLE ... SET TBLPROPERTIES.  **Bronze → 30 Days**  sql  CopyEdit  ALTER TABLE sbk\_customer360.bronze.web\_events  SET TBLPROPERTIES ('delta.deletedFileRetentionDuration' = '30 days');  **Silver → 90 Days**  sql  CopyEdit  ALTER TABLE sbk\_customer360.silver.crm\_clean  SET TBLPROPERTIES ('delta.deletedFileRetentionDuration' = '90 days');  This controls how long **old versions and deleted files** are kept for Time Travel and vacuum safety.  **2. Notebook Archival to Git (Monthly)**  Automate via CI/CD or a scheduled GitHub Action.  **🔧 Sample GitHub Workflow (.github/workflows/archive\_notebooks.yml)**  yaml  CopyEdit  name: Archive Notebooks Monthly  on:  schedule:  - cron: '0 1 1 \* \*' # Runs on the 1st day of every month at 1 AM  workflow\_dispatch:  jobs:  archive:  runs-on: ubuntu-latest  steps:  - name: Checkout notebooks  uses: actions/checkout@v3  - name: Commit archived notebooks  run: |  git config user.name "databricks-bot"  git config user.email "bot@databricks.com"  git add src/notebooks/  git commit -m "Monthly archive of notebooks - $(date +'%Y-%m-%d')" || echo "No changes"  git push origin main  Ensure notebooks are in .py or .dbc format via %notebook save or Repos sync.  **3. Job Results and Log Expiry – 14 Days**  This is configured in **Databricks Admin Console**, but here's how to set it via **workspace APIs** or Terraform.  **Terraform (if using automation)**  hcl  CopyEdit  resource "databricks\_workspace\_conf" "job\_retention" {  custom\_config = {  "spark.databricks.jobResultsRetentionInDays" = "14"  }  }  Or manually go to **Admin Console → Workspace Settings** → set:   * Job Results Retention = **14 days**   **Summary Table**   |  |  | | --- | --- | | Item | Code / Action | | Delta Bronze Retention (30 days) | ALTER TABLE ... SET TBLPROPERTIES | | Delta Silver Retention (90 days) | ALTER TABLE ... SET TBLPROPERTIES | | Notebook Archival to Git | GitHub Actions cron job | | Job Logs Expiry (14 days) | Admin Console or databricks\_workspace\_conf via Terraform | |

### 12. Documentation Standards

* Markdown files in /docs/
* Content:
  + System overview (Customer360 architecture)
  + Customer entity schema
  + Job ownership and contact
  + External data contracts (CRM, Support API, Web tracking)

**Documentation Standards**

|  |  |
| --- | --- |
| Location | Description |
| /docs/ | Centralized location for project documentation in Markdown format (.md). |
| System Overview | Includes high-level architecture of the Customer360 platform, data flow, and environment separation. |
| Entity Schemas | Defines structure, keys, and relationships for core entities (e.g., customers, transactions, interactions). |
| Job Ownership | Maps each notebook or job to an owner with email/contact and escalation paths. |
| Data Contracts | Lists external dependencies such as CRM systems, support APIs, and web tracking inputs, including refresh frequency and contact points. |

This ensures:

* Easy onboarding for new engineers and analysts
* Clear accountability and traceability
* Alignment with upstream and downstream data providers

### 

### 13. Tools & Integrations

|  |  |
| --- | --- |
| Tool | Purpose |
| Unity Catalog | Access control and data cataloging |
| GitHub | Version control and automated CI/CD |
| MLflow | Track experiments for churn model |
| Terraform | Automate workspace and cluster setup |
| Azure Monitor | Monitor job performance and SLA violations |

### 14. Summary

This folder structure and governance model supports Customer360 initiative by enabling scalable, secure, and maintainable workflows across data ingestion, transformation, and analytics. It provides unified access control, automation readiness, and data transparency for all customer data stakeholders.

Appendix:

The **Customer360 Databricks Folder and Artifact TDD Template** helps in multiple ways depending on your role—whether you’re a **Data Engineer, Solution Architect, or Platform Admin**.

**Who is it for?**

* **Data Engineers**: Standardize where to place notebooks, pipelines, and tests.
* **ML Engineers**: Isolate experimental work in dev, productionize workflows cleanly.
* **Solution Architects**: Enforce scalable workspace design across teams.
* **Platform Admins**: Set up access controls, automation, and monitoring easily.
* **Project Managers**: Align technical team efforts under clear structure.

**How This Template Helps**

|  |  |
| --- | --- |
| Category | How the Template Helps |
| Folder Organization | Provides a **repeatable and scalable folder structure** across dev/test/prod. |
| Artifact Naming | Avoids ambiguity by standardizing names for jobs, notebooks, tables, and pipelines. |
| Governance | Integrates Unity Catalog for **RBAC**, sensitive data masking, and schema control. |
| CI/CD | Lays out Git repo layout and versioning practices for seamless **collaborative dev**. |
| DLT and Pipelines | Prepares teams for building **Delta Live Table pipelines** with quality expectations. |
| Job and Resource Management | Includes best practices for **job tagging**, folder separation, and cleanup retention. |
| Documentation Standards | Promotes clear architecture diagrams, data dictionaries, and team accountability. |
| Infrastructure as Code | Prepares you for automating the environment using **Terraform or similar tools**. |

**Example in Use : SBK Customer360**

In the SBK example:

* Data from CRM, Web, and Support flows into a structured folder setup.
* Each pipeline is isolated and named clearly.
* Analysts can query gold tables confidently, knowing the lineage and ownership.
* New engineers onboard faster with /docs/ containing architecture and notebook purpose.