ODS\_SIT\_AUTOMATION\_POC/

│

├── notebooks/

│ └── UKS\_SIT\_DW\_E2E/

│ ├── ODS\_BRONZE\_VALIDATION\_ESSENTIAL.ipynb # Core validation script

│ ├── DLT\_REFRESH\_PIPELINE.ipynb # Trigger and refresh DLT

│ ├── DDL\_COMPARISION\_ORA\_ODS\_S3.ipynb # DDL schema comparison

│ ├── ORA\_READREPLICA\_SCHEMA\_CHECK.ipynb # Source system validation

│ └── REFERENCE\_FILE/

│ ├── exclusin\_list.csv

│ ├── ora\_schema\_file.csv

│ └── readreplica\_schema\_file.csv

│

├── Release\_7\_RunNoteBook\_Deploy.ipynb # CI Trigger entry notebook

├── Create\_Databricks\_Job.ipynb # Dynamic job generator

├── azure-pipelines.yml # CI/CD pipeline config

├── EDM Scripts.sql # Source scripts (optional)

├── IDW\_Load\_Test.sql # Data setup for validation

├── README.md # Repository overview

## ****17.1 Automation Components Overview****

The automation framework is designed to run as a **modular, version-controlled system** integrated with Azure DevOps for deployment, **Databricks Workflows** for orchestration, and **Delta Live Tables (DLT)** for result storage and dashboard generation.

## ****17.2 Azure DevOps Folder Structure****

Project Folder: ODS\_SIT\_AUTOMATION\_POC

pgsql

CopyEdit

ODS\_SIT\_AUTOMATION\_POC/

│

├── notebooks/

│ ├── UKS\_SIT\_DW\_E2E/

│ │ ├── ODS\_BRONZE\_VALIDATION\_ESSENTIAL.ipynb # Core data validation logic

│ │ ├── DLT\_REFRESH\_PIPELINE.ipynb # Refreshes DLT layer

│ │ ├── DDL\_COMPARISION\_ORA\_ODS\_S3.ipynb # Schema diff between Oracle, ODS, and CDA

│ │ ├── ORA\_READREPLICA\_SCHEMA\_CHECK.ipynb # Validates schema between Oracle and Read Replica

│ │ └── REFERENCE\_FILE/

│ │ ├── exclusion\_list.csv # Tables to skip from automation

│ │ ├── ora\_schema\_file.csv # Baseline Oracle schema

│ │ ├── readreplica\_schema\_file.csv # Baseline Read Replica schema

│

├── Release\_7\_RunNoteBook\_Deploy.ipynb # Trigger notebook for CI/CD

├── Create\_Databricks\_Job.ipynb # Creates or updates Databricks job/workflow

├── azure-pipelines.yml # DevOps CI/CD definition

├── EDM Scripts.sql, IDW\_Load\_Test.sql # Supporting SQL scripts

├── README.md # Usage guide and execution instructions

## ****17.3 CI/CD Deployment Workflow (Azure DevOps)****

### 🔁 **Step-by-Step DevOps Flow**

1. **Commit to Git Repo**: Developer commits updated validation notebooks to notebooks/UKS\_SIT\_DW\_E2E/.
2. **CI/CD Trigger via Pipeline (azure-pipelines.yml)**:
   * Validates the commit
   * Deploys notebooks to **Development** Databricks workspace
3. **Trigger Notebook (Release\_7\_RunNoteBook\_Deploy.ipynb)**:
   * Deploys target notebooks into **Test** workspace upon approval
4. **Workflow Definition (Create\_Databricks\_Job.ipynb)**:
   * Dynamically creates or updates **Databricks Workflows** for Test execution

✅ Deployment to **Pre-Prod/Prod** requires stakeholder approval and follows the same CI/CD route via **Release Pipelines**.

## ****17.4 Test Environment Execution Flow****

### 🔁 **Workflow Stages**

1. **Workflow Schedules**:
   * Runs validation notebooks via **Databricks Workflows** daily or on-demand
   * Covers: DDL check, Bronze validation, schema checks
2. **DLT Pipeline (DLT\_REFRESH\_PIPELINE.ipynb)**:
   * Writes test results to **Delta Live Tables**
   * Handles cleanup and refresh operations
3. **Validation Result Tables**:
   * Structured into:
     + ods\_test\_result\_main (test case summary)
     + ods\_test\_result\_detail (column-by-column detail)
     + ods\_schema\_diff\_table (DDL and constraint mismatches)
4. **Dashboard Integration**:
   * Dashboards are linked to DLT views
   * Show test pass/fail, execution time, table-level issues
   * Auto-refreshes based on workflow result updates

## ****17.5 Dashboard Functionality & Usability****

* **Views Provided**:
  + Summary by ingestion type (Read Replica vs. CDA)
  + Failure reason per table or test type
  + Schema difference heatmap
  + Ingestion time comparison: ADF vs Databricks
* **Usability Highlights**:
  + Filter by date, layer, table name
  + Click-through to raw details from summary
  + Easy export (CSV/Excel) for reporting
  + Ideal for stakeholder demos and QA sign-offs

## ****17.6 Live (Production) Execution Strategy****

* Once the Test workflows and validation are successful, the setup is:
  1. **Copied from Test to Live** using **Data Engineering’s approved migration scripts**
  2. Configurations (e.g., cluster, table path, layer alias) are **updated to Prod context**
  3. Workflows run in **read-only mode**, focusing on:
     + Confirming delta ingestion
     + Schema integrity
     + File generation in outbound layers

## ****17.7 Configuration Setup****

| **Configuration Area** | **Details** |
| --- | --- |
| **Cluster Setup** | Medium-size autoscaling cluster (8–16 cores, 64–128 GB RAM), tag-based for environment isolation |
| **Databricks Secrets** | Used to store JDBC credentials, S3 access keys, and mount details |
| **Parameterization** | Notebooks accept parameters like env, source\_type, table\_group |
| **Table Exclusion** | Driven by exclusion\_list.csv |
| **Schema Baselines** | Stored in CSVs and dynamically loaded into comparison logic |

## ****17.8 Team Communication & Outlook Integration****

* **Email Alerts** (in progress):
  + Sent via Outlook when:
    - Workflow fails
    - Schema mismatch is detected
    - Daily dashboard summary is ready
* **Stakeholder Updates**:
  + Weekly Excel/PowerPoint extracted from dashboards
  + Shared via Teams or Confluence

## ✅ Summary

This modular, CI/CD-enabled automation setup ensures:

* **Low-code onboarding**
* **Seamless workflow portability** from Test to Live
* **Highly visible dashboard reporting**
* **Tight integration with DevOps and Data Engineering processes**

## Guidewire Data Migration Automation – Confluence Documentation

### Overview

This Confluence page outlines the automation components, architecture, workflows, and orchestration setup used for validating the Guidewire data migration into Azure Data Lake. The process is implemented using Azure Databricks, Azure DevOps, Delta Live Tables (DLT), and connected dashboards for visibility.

### 1. Automation Code Repository (Azure DevOps)

**Repository Name:** ODS\_SIT\_AUTOMATION\_POC

**Folder Structure:**

ODS\_SIT\_AUTOMATION\_POC

│── notebooks/

│ │── UKS\_SIT\_DW\_E2E/

│ │ │── ODS\_BRONZE\_VALIDATION\_ESSENTIAL.ipynb # Core data validation logic

│ │ │── DLT\_REFRESH\_PIPELINE.ipynb # Refreshes DLT pipeline

│ │ │── REFERENCE\_FILE/

│ │ │ │── table\_list.csv # List of tables and attributes

│ │── Release\_7\_RunNoteBook\_Deploy.ipynb # CI/CD trigger notebook

│ │── Create\_Databricks\_Job.ipynb # Automates job creation in Databricks

│── azure-pipelines.yml # Defines the build/release pipeline

│── EDM Scripts.sql # Reference DDL and transformation rules

│── IDW\_Load\_Test.sql # Test data insertion script

│── README.md # Developer/QA setup instructions

### 2. CI/CD Pipeline Strategy

* **Code push to Dev branch** triggers notebook validation and internal testing.
* **Merge to Main/Release branch** triggers Azure DevOps pipeline:
  + Deploys notebooks to **Dev environment** for CI validation.
  + On approval, deploys validated notebooks to **Test environment**.
* **Live Migration:**
  + After successful testing and approval, Data Engineering team uses their **copy mechanism** to replicate Test workflows and assets into the **Live/Production environment**, mimicking the same configuration.

### 3. Test Environment Setup & Workflow

#### **Workflow Components:**

1. **ODS\_BRONZE\_VALIDATION\_ESSENTIAL.ipynb**
   * Validates data between staging sources (Oracle, Read Replica, CDA) and Bronze layer.
   * Checks include schema, row counts, data match, SCDs, duplicates.
2. **DLT\_REFRESH\_PIPELINE.ipynb**
   * Refreshes Delta Live Table pipelines that store validation results.
3. **Create\_Databricks\_Job.ipynb**
   * Automates job creation and configuration in Databricks.

#### **DLT Integration:**

* Test results (row counts, mismatches, anomalies, SCD discrepancies) are written to **Delta Live Tables**.
* DLT views are built on top for downstream reporting.

#### **Dashboard Integration:**

* Dashboards are built on top of DLT views using Databricks Dashboard feature.
* Results are visualized per ingestion path (Read Replica, CDA), per table, and per rule.
* Easy filtering by environment, test cycle, and date.

### 4. Orchestration & Scheduling

* **Databricks Workflows** are used to:
  + Schedule data validation notebooks (ODS\_BRONZE\_VALIDATION\_ESSENTIAL).
  + Trigger DLT refresh pipelines.
  + Run reference lookups for dynamic metadata.
* Schedules can be:
  + Daily (Post-ingestion cycle)
  + Manual trigger (for revalidation)

### 5. Test Result Views

|  |  |
| --- | --- |
| **View Name** | **Description** |
| validation\_summary\_view | High-level pass/fail per table and rule |
| row\_mismatch\_detail\_view | Detailed mismatched rows (field-level) |
| schema\_comparison\_view | Expected vs actual schema for each table |
| duplicate\_check\_view | Duplicate records identified by primary/business key |

### 6. Test Result Metadata

Each test result is stored with the following metadata fields:

* table\_name
* validation\_type (e.g., schema, count, accuracy)
* source\_system (Oracle, Read Replica, CDA)
* execution\_date
* pass\_flag
* record\_count / mismatch\_count
* execution\_time

### 7. Dashboard Functionality & Usability

* **Filters:** by date, validation type, ingestion source, and status.
* **Charts:** record count mismatches, schema validation summaries, SCD vs actual.
* **Widgets:** success/fail rate by layer, execution time comparison.
* **Ease of Use:**
  + Designed for QA, DevOps, and Product stakeholders.
  + Drill-down from summary to row-level issues.
  + Export to CSV for offline reporting or JIRA linking.

### 8. Outlook & Team Integration

* Ongoing effort to connect results to **Outlook Alerts / Teams Channels**.
* Post-test summary (daily or weekly) to be auto-emailed to key stakeholders.
* Jira integration remains manual (due to firewall), but export-ready logs are aligned with test cases.

### 9. Configuration Management

* **Cluster Policies** defined by environment (Dev, Test, Prod)
* **DLT Configs:** include pipeline name, cluster, table path, schema mode
* **Pipeline Parameters**: allow test cycle, environment name, and table list to be passed dynamically

## ****Section 17: Automation Components and Step-by-Step Workflow****

This section provides a detailed breakdown of all automation components, their structure within Azure DevOps, deployment pipeline workflows, and Databricks validation orchestration steps.

### 📁 **17.1 Azure DevOps Repository Structure**

nginx

CopyEdit

ODS\_SIT\_AUTOMATION\_POC

│── notebooks/

│ │── UKS\_SIT\_DW\_E2E/

│ │ │── ODS\_BRONZE\_VALIDATION\_ESSENTIAL.ipynb # Core data validation notebook

│ │ │── DLT\_REFRESH\_PIPELINE.ipynb # Delta Live Table pipeline notebook

│ │ │── REFERENCE\_FILE/

│ │ │ │── table\_list.csv # Table lists approved for validation

│ │── Release\_7\_RunNoteBook\_Deploy.ipynb # Notebook trigger for pipeline deployment

│ │── Create\_Databricks\_Job.ipynb # Automated job creation in Databricks

│── azure-pipelines.yml # CI/CD deployment pipeline definition

│── EDM Scripts.sql # EDM related scripts for validation

│── IDW\_Load\_Test.sql # Sample load and ingestion test scripts

│── README.md # Project overview documentation

### 🚀 **17.2 Azure DevOps Pipeline Workflow**

**Pipeline**: Managed via **azure-pipelines.yml** to orchestrate notebook deployments across environments.

| **Step** | **Action** | **Environment** | **Approval Required?** |
| --- | --- | --- | --- |
| 1 | Commit & Push notebooks/files to Azure DevOps Git repo | Development | No |
| 2 | Trigger Azure DevOps pipeline (CI/CD via azure-pipelines.yml) | Development → Test | Yes (QA Lead) |
| 3 | Deploy notebooks to Databricks Test workspace | Test | Yes (QA Lead Approval via Azure DevOps) |
| 4 | Execute validation workflows in Databricks | Test | No |
| 5 | Validation passed; trigger deployment from Test → Pre-Prod/Prod | Pre-Prod → Prod | Yes (Release Manager approval required via Azure DevOps) |

### ⚙️ **17.3 Databricks Environment Workflow (Test Environment)**

Once notebooks are deployed in Databricks test workspace:

* **Step 1: Create Databricks Job**
  + Execute Create\_Databricks\_Job.ipynb
  + Configure notebook execution parameters and clusters (compute, storage, IAM role)
* **Step 2: Refresh Delta Live Table (DLT) Pipeline**
  + Execute DLT\_REFRESH\_PIPELINE.ipynb
  + Establish schema and table structures for validation results
* **Step 3: Execute Core Validation Notebook**
  + Execute ODS\_BRONZE\_VALIDATION\_ESSENTIAL.ipynb
  + Perform schema validation, data accuracy, duplicates, count checks, etc.
* **Step 4: Store Results in Delta Live Tables**
  + Validation results are written automatically to DLT for structured storage and management
* **Step 5: Dashboard Integration**
  + Dashboards built directly on Delta Live Tables for reporting metrics
  + Results include schema diffs, row counts, mismatches, duplicates, and performance metrics
* **Step 6: Teams and Outlook Integration (in progress)**
  + Results notification and summary automatically pushed to Teams and Outlook for quick review (future-state setup)

### 📊 **17.4 Production Workflow (Post-Approval)**

Following successful test validation and release manager approval:

* Approved notebooks and pipeline definitions are pushed from Test → Production Databricks via Azure DevOps release pipeline.
* Execute production workflows (read-only execution):
  + DLT\_REFRESH\_PIPELINE.ipynb (refresh validation tables)
  + ODS\_BRONZE\_VALIDATION\_ESSENTIAL.ipynb (perform validation)
* Production Delta tables updated with execution results.
* Dashboards reflect live production validation metrics.

### **17.6 Dashboard Reporting & Integration**

**Dashboard Setup in Databricks:**

* Dashboards are linked directly to the DLT tables.
* Visualize test results: Pass/Fail, discrepancies, runtime metrics.
* Integration with **Teams and Outlook** in progress to provide automated notifications.

### 📍 **17.7 Execution Checklist (Environment-wise)**

| **Environment** | **Tasks** |
| --- | --- |
| **Development** | Validate CI/CD pipeline integration, notebook structure, and deployment scripts |
| **Test** | Execute full DLT pipeline, comprehensive validation, dashboard reporting setup |
| **Pre-Prod/Production** | Approved workflow deployments only, execution monitoring, restricted access |

### **Next Steps & Recommendations**

* Complete Outlook and Teams notification integration for automated test summaries.
* Finalize approval gates in Azure DevOps pipeline (manual approval process).
* Regularly review and update table\_list.csv to reflect current test scope.

## ****Section 17: Automation Components, Configuration, and Step-by-Step Workflow****

This section outlines the detailed components, configurations, and step-by-step execution processes that constitute the automation testing framework for validating Guidewire data ingestion methods.

### 🔧 **17.1 Automation Components**

| **Component** | **Description** | **Technology/Tools** |
| --- | --- | --- |
| **Azure DevOps Repository** | Version-controlled storage for automation scripts, notebooks, and pipeline YAML definitions | Azure Git Repo |
| **Azure DevOps Pipeline** | CI/CD pipeline that deploys notebooks, scripts, and configurations to Databricks environments | Azure DevOps (YAML Pipelines) |
| **Databricks Workspace** | Execution environment for automation validation scripts and workflows | Azure Databricks |
| **Databricks Notebooks** | Validation logic written in PySpark to run schema checks, count checks, data accuracy, etc. | Python, PySpark |
| **Databricks Workflows** | Orchestrates notebook execution, scheduled to run validations regularly | Databricks Jobs & Workflow Scheduler |
| **Delta Live Tables (DLT)** | Stores validation results, intermediate results, and logs | Delta Lake (DLT) |
| **Databricks Dashboards** | Visualizes validation metrics, status, and execution outcomes | Databricks SQL Dashboards |
| **Integration & Reporting** | Manual reporting into JIRA and automated reports sent to MS Teams/Outlook | Databricks, Excel, Outlook, Teams |

### ⚙️ **17.2 Azure DevOps Configuration (Files & Repos)**

| **Repo Structure** | **Content/Files** |
| --- | --- |
| /notebooks/ | PySpark validation scripts |
| /pipelines/ | YAML definitions for deployment pipeline |
| /configs/ | JSON/YAML config files per environment |
| /sql/ | Generated SQL scripts for validation |
| /docs/ | Documentation (setup, deployment guide, user manual) |

**Sample Structure:**

pgsql

CopyEdit

automation-framework-repo/

├── notebooks/

│ ├── bronze\_validation.ipynb

│ ├── silver\_validation.ipynb

│ ├── gold\_validation.ipynb

│ └── performance\_check.ipynb

├── pipelines/

│ ├── deploy-to-dev.yml

│ ├── deploy-to-test.yml

│ └── deploy-to-prod.yml

├── configs/

│ ├── dev-config.json

│ ├── test-config.json

│ └── prod-config.json

├── sql/

│ └── validation\_queries.sql

└── docs/

├── deployment\_steps.md

└── README.md

### 🔄 **17.3 CI/CD Deployment Workflow (Azure DevOps Pipelines)**

**Step-by-step pipeline workflow:**

**Development (DEV) Deployment:**

* Developers commit notebooks/config changes into a feature branch.
* Code review and merge into dev branch.
* Azure DevOps pipeline triggered:
  1. Checkout and validate files.
  2. Deploy notebooks & configurations to Databricks DEV environment.
  3. Run integration tests for deployment confirmation.

**Test Environment Deployment:**

* Merge dev → main branch triggers pipeline:
  1. Checkout validated notebooks/config from Azure Git.
  2. Deploy notebooks to Databricks TEST workspace.
  3. Deploy workflow and DLT configurations.
  4. Manual or automatic approval gates control execution.

**Production Deployment:**

* Release branch created from main.
* Approval required (QA/management).
* Pipeline execution:
  1. Deploy notebooks to PROD workspace.
  2. Configure DLT pipelines and Databricks Workflows.
  3. Lock configurations post-deployment.

### 🧩 **17.4 Databricks Workflow & DLT Pipeline Execution**

**Workflow setup (Test & Production):**

* Defined using Databricks Workflow Scheduler.
* Scheduled or event-driven (ingestion completion).
* Execution sequence:
  1. Run **DDL validation notebook** to confirm table structure.
  2. Run **data validation notebooks** for schema, count, accuracy.
  3. Run **performance check notebook** to benchmark.
  4. Results written to **Delta Live Tables (DLT)**.

**DLT Pipeline configuration:**

* **Input Layer:** Read from staging sources.
* **Validation Layer:** Intermediate and final result tables.
* **Reporting Layer:** Aggregated view tables for dashboard reporting.

### 📊 **17.5 Dashboard Integration & Monitoring**

* Dashboards created in Databricks SQL, visualizing:
  + Validation statuses by layer (Bronze/Silver/Gold).
  + Error counts, mismatch counts.
  + Run-times per notebook and per test.
* Dashboard refreshed automatically post-workflow execution.

### ✉️ **17.6 Reporting & Integration with Teams/Outlook**

* Automated export of validation results (Excel/CSV) for manual JIRA updates.
* Dashboard links shared through automated emails (Outlook) and MS Teams alerts.
* Escalation workflow defined for critical validation failures.

### 📋 **17.7 Step-by-Step Automation Execution (Daily Run Example)**

1. **Pre-Check (Manual)**:
   * Staging data availability verified.
   * Cluster resources verified.
2. **Workflow Trigger** (Automated Databricks):
   * Execute workflow job.
3. **Execution Steps** (Automated):
   * Run schema validation (Notebook).
   * Run record count validation (Notebook).
   * Run detailed data accuracy check (Notebook).
   * Performance benchmark notebook execution.
4. **Post-Execution**:
   * Results stored in DLT tables.
   * Dashboards automatically refreshed.
   * Status notification sent via Teams/Outlook.
5. **Manual Reporting**:
   * QA Coordinator exports results for manual JIRA update.
   * Defect logging and follow-up as needed.

### ✅ **17.8 Prerequisites for Configuration**

* Databricks workspace configured with required clusters and VPN for S3 access.
* Azure DevOps repository created, structured as per above guidelines.
* CI/CD pipeline configured and permissions provided.
* Access and permissions to configure Databricks Workflows, Dashboards, and DLT.