π - Recon

Version: 01
Author: Pibythree Developers

Table of Contents

1. Introduction

This document outlines the requirements, architecture, and implementation details for the **Pi-Recon** solution. The objective of this initiative is to deliver a secure, scalable, and reliable data reconciliation framework — **distributed as a Snowflake Native App** — that enables seamless comparison and validation of data between source and target database systems within the consumer's Snowflake account.

The document serves as a single source of truth for all stakeholders involved in the project, including business users, data engineers, solution architects, testers, and project managers. It captures both functional and non-functional requirements, along with detailed backend and frontend implementation approaches within the Snowflake Native App Framework.

The Pi-Recon solution leverages the **Snowflake Native App distribution model** and high-performance capabilities such as **Snowpark Container Services (SPCS)**, **Snowflake Image Repositories**, and **Reference Binding** for secure secrets management and external access. Unlike headless ingestion apps, Pi-Recon provides a **full web-based user interface** accessible directly from Snowsight, enabling consumers to configure projects, map objects, execute reconciliation, and view results — all without leaving the Snowflake ecosystem.

This initiative is part of the broader **Pi-Snow** roadmap and is designed to reduce manual data validation effort, improve data quality visibility, and provide auditable reconciliation reports through a packaged application model.

2. Key Considerations

This section outlines the foundational elements that influence the scope, planning, and execution of the Pi-Recon project, including scope boundaries, assumptions, dependencies, and risks.

2.1 Scope

The scope of this project includes the following:

- Row-level data reconciliation between source and target databases
- Column-level aggregate comparison (sum, count, mean, median)
- Multi-database connectivity: MySQL, Microsoft SQL Server (MSSQL), and Snowflake
- Metadata extraction and discovery from source and target systems
- Object selection, table mapping, and column mapping via web UI
- Statistical sampling for large datasets
- Project-based access control with multi-user support
- Encrypted credential storage for external database connections
- Reconciliation result storage with variance analysis
- Web-based UI accessible via Snowflake Native App endpoint

2.2 Out of Scope

The following items are not included in the current scope:

- Data ingestion, migration, or transformation between systems
- Automated remediation or data correction based on mismatches
- Real-time or streaming reconciliation
- Scheduling of reconciliation jobs (manual trigger via UI)

2.3 Assumptions

Sr. No	Description	Implications, if not met
1	Active source/target database credentials available to consumer	Unable to connect and extract metadata
2	Snowflake account with Enterprise edition and SPCS enabled	Application cannot be deployed or executed
3	Required Snowflake roles and permissions are available (ACCOUNTADMIN or equivalent)	Objects and services cannot be created
4	Network access from Snowflake SPCS to external database endpoints is allowed	External connectivity will fail
5	Source and target databases have compatible schemas for comparison	Reconciliation results may be incomplete

Sr. No	Description	Implications, if not met
6	MFA-enabled Snowflake accounts use key-pair authentication	Password-based connections will timeout

2.4 Dependencies

Sr. No	Description	Owner	Implications, if not met
1	External database credentials (MySQL, MSSQL, or external Snowflake)	Consumer Database Admin	Data extraction and comparison cannot be initiated
2	Application installation and granting of account-level privileges	Consumer AccountAdmin	App cannot provision infrastructure or run services
3	Security approval and Reference Binding for Secrets and External Access	Consumer AccountAdmin	Application is blocked from connecting to external databases
4	Compute pool and warehouse availability	Consumer AccountAdmin	Service execution delays or failures
5	Network rules permitting egress to source/target database hosts	Consumer Network Admin	Connection timeouts during metadata extraction or reconciliation

2.5 Risks

Sr. No	Description	Mitigation Strategy
1	Credential exposure or misconfiguration	Use Snowflake Secrets with Reference Binding to ensure encryption keys remain within the consumer account boundary
2	Consumer region incompatibility for SPCS	Validate Snowpark Container Service availability in the target consumer region prior to app installation
3	Large dataset memory exhaustion	Statistical sampling limits data pulled into memory; configurable confidence level and error margin
4	External database network timeouts	Clear documentation of required network rules; error messages guide consumer to update EAI
5	Encryption key rotation breaks stored credentials	Document key lifecycle; warn consumers to re-encrypt connections after key changes

3. Business Feature Details

This section outlines the key business functionalities delivered by the Pi-Recon solution. Each feature is aligned with business objectives to ensure reliable, auditable, and actionable data reconciliation.

3.1 Business Process Flow

The Pi-Recon solution enables end-to-end data validation between any source and target database system:

1. Connect — Consumer configures connections to source and target databases
2. Discover — Metadata (schemas, tables, columns) is extracted from connected systems
3. Map — Source tables/columns are mapped to target tables/columns
4. Configure — Validation rules (key columns, aggregations, ignore flags) are defined
5. Reconcile — Row-level and column-level comparisons are executed
6. Report — Results are stored in Snowflake with match/mismatch classification and variance metrics

3.2 Reconciliation Capabilities

The solution supports two levels of data validation:

Row-Level Reconciliation

- Record-by-record comparison between source and target using key columns
- Identifies: MATCH, MISMATCH, SOURCE_ONLY, TARGET_ONLY
- Reports which specific column values differ for each mismatched record

Column-Level Reconciliation

- Aggregate comparison using configurable metrics: SUM, COUNT, MEAN, MEDIAN, AVERAGE
- Reports variance between source and target aggregates
- Suitable for high-level data parity verification without full row scan

3.3 Statistical Sampling

For large datasets, Pi-Recon implements statistical sampling to reduce execution time while maintaining confidence in results:

- **Configurable Parameters:** Confidence Level (default 95%), Error Margin (default 5%)
- **Sampling Engine:** Uses statistics-based calculations to determine optimal sample size
- **Application:** When enabled, only a statistically representative subset of records is compared

3.4 RBAC & UBAC

Role Based Access Control (RBAC) and User Based Access Control (UBAC) are implemented to ensure secure and governed access to reconciliation data.

RBAC — Application Roles

- **APP_ADMIN:** Full access to all application features and functions, including creating new connections
- **APP_USER:** Full access to all application features and functions **except creating new connections**
- **APP_VIEWER:** Read-only access, limited to viewing reconciliation results only

UBAC — User Identification

- In SPCS, user identity is derived from the Snowflake ingress session (no separate login required)

3.5 Monitoring & Operational Visibility

The solution includes monitoring and logging capabilities:

- **Run Logging:** Every reconciliation execution is recorded with start time, end time, duration, record counts, and match/mismatch statistics
- **Error Tracking:** Failed reconciliation runs capture detailed error messages for troubleshooting
- **Result Persistence:** All row-level and column-level results are stored for historical analysis
- **Reporting Views:** Pre-built views (V_RECENT_RUNS, V_MISMATCH_SUMMARY) provide instant operational visibility

4. Technical Feature Details

This section outlines the technical interpretation of business features. It translates business needs into technical components, ensuring alignment between business goals and system capabilities within the Snowflake Native App Framework.

4.1 Technical Architecture

The solution follows a Snowflake Native App architecture with a containerized web application:

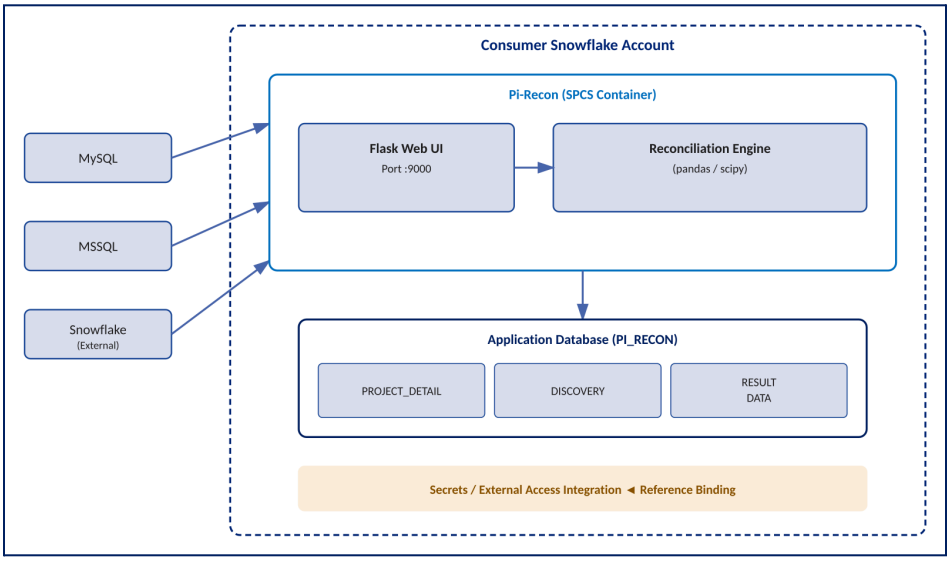

Fig 1 — Pi-Recon Architecture

- **Web UI Layer:** Flask application running in SPCS, accessible via Snowsight
- **Reconciliation Engine:** Python-based comparison logic using pandas and scipy
- **External Connectivity:** Outbound connections to MySQL, MSSQL, External Snowflake via EAI
- **Secure Credential Management:** Encryption keys from Snowflake Secrets; connection passwords encrypted at rest in application tables
- **Result Storage:** All results stored in the application's Snowflake database for persistence

4.2 Web Application Layer (Flask UI)

Pi-Recon exposes a full web-based interface via SPCS public endpoint:

- **Framework:** Python Flask (v3.0.3) with Jinja2 templating
- **Port:** 9000 (exposed via SPCS endpoint)
- **Access:** Consumers click "Open" in Snowsight to access the UI (already authenticated via Snowflake)
- **Client-Side:** Tailwind CSS, jQuery, Axios for AJAX, SheetJS for Excel export
- **Session Management:** Flask signed cookies with SECRET_KEY from Snowflake Secret

- **Authentication:** No separate login — user identity derived from SPCS ingress (Snowflake session)

UI Capabilities

- Connection CRUD with live test
- Project creation with source/target assignment
- Background metadata extraction with polling
- Interactive table and column mapping
- One-click reconciliation execution
- Results dashboard with drill-down and Excel export

4.3 Multi-Database Connectivity

Pi-Recon supports outbound connections to three database systems:

Database	Connector	Default Port	Auth Methods
MySQL	mysql-connector-python	3306	Username/Password
MSSQL	pymssql	1433	Username/Password
Snowflake	snowflake-connector-python	443	Password, Key-Pair (RSA)

All connections are established within the SPCS container using the consumer-provided External Access Integration for egress.

4.4 RBAC & UBAC

Access control is implemented using Application Roles defined within the Native App:

- **Application Roles:** Defines internal roles (APP_ADMIN, APP_USER, APP_VIEWER) that control access to schemas, tables, procedures, and services
- **Consumer Mapping:** The consumer maps these Application Roles to their internal Snowflake roles using RBAC
- **Data Isolation:** Connection credentials are encrypted and stored in restricted tables accessible only through the application's internal logic

4.5 Logging & Monitoring

- **Run Logging:** Every reconciliation execution is captured in RESULT.RUN_LOG with status, duration, record counts, and error messages
- **Result Detail:** Row-level mismatches stored in RESULT.ROW_LEVEL_RESULT; column-level variances in RESULT.COLUMN_LEVEL_RESULT
- **Container Logs:** SPCS container stdout/stderr available via Snowsight Monitoring > Services & Logs
- **Error Handling:** Application-level exceptions are caught and stored with stack traces for troubleshooting

4.6 Secrets & Secure Access Management

This solution implements secure credential management and controls external connectivity without exposing sensitive information in the application code or to the application provider.

4.6.1 Connection Credential Storage

Database credentials entered through the Pi-Recon UI are stored in encrypted form inside the application database.

Credential storage table: PROJECT_DETAIL.CONNECTIONS

Relevant columns:

- PASSWORD_ENCRYPTED: encrypted password-based credentials
- PRIVATE_KEY_ENCRYPTED: encrypted private key content for Snowflake key-pair authentication
- PRIVATE_KEY_NAME: uploaded private key file name
- AUTH_TYPE: authentication mode such as password, PAT, or key_pair

Credentials are used only by the Pi-Recon service for connection testing, metadata extraction, and reconciliation execution.

4.6.2 Network Rule Configuration

Network rules are defined by the consumer to control outbound connectivity:

- **Egress Control:** Network rules allow outbound access only to approved database hosts
- **Default Whitelist:** *.snowflakecomputing.com:443 for external Snowflake accounts
- **Consumer-Added Hosts:** MySQL (port 3306) and MSSQL (port 1433) hosts are added by the consumer post-installation
- **Granular Security:** Network rules ensure the container can only communicate with verified endpoints

4.6.3 External Access Integration

The EAI acts as the secure bridge enabling outbound communication:

- **Reference Binding:** The application requests access to a consumer-defined EAI through Reference Binding
- **Security Association:** The EAI links approved Network Rules to the authorized Pi-Recon service.
- **Restricted Execution:** Only the authorized SPCS service within the application is permitted to use the integration

4.6.4 Provider Isolation

The application provider does not have access to consumer-entered database credentials, source data, target data, or reconciliation results.

All project configuration, metadata, mappings, and reconciliation results remain inside the consumer Snowflake account.

4.7 Authentication Mechanisms

4.7.1 Snowflake Password Authentication

Standard username/password authentication for external Snowflake accounts without MFA.

4.7.2 Snowflake Key-Pair Authentication

RSA key-pair (PKCS8 DER format) authentication. Private key is encrypted and stored in the CONNECTIONS table. Recommended for production and MFA-enabled accounts.

4.7.3 MySQL & MSSQL Authentication

Standard username/password authentication over TCP connections via EAI-controlled egress.

4.7.4 MFA-Enabled Account Handling

Snowflake accounts with MFA enforcement cannot use password authentication from a headless container (no browser for push notification). Consumers must use **key-pair authentication** for MFA-enabled accounts. This is clearly documented in the setup guide.

5. Technical Details – Frontend

5.1 Page Flow & Navigation

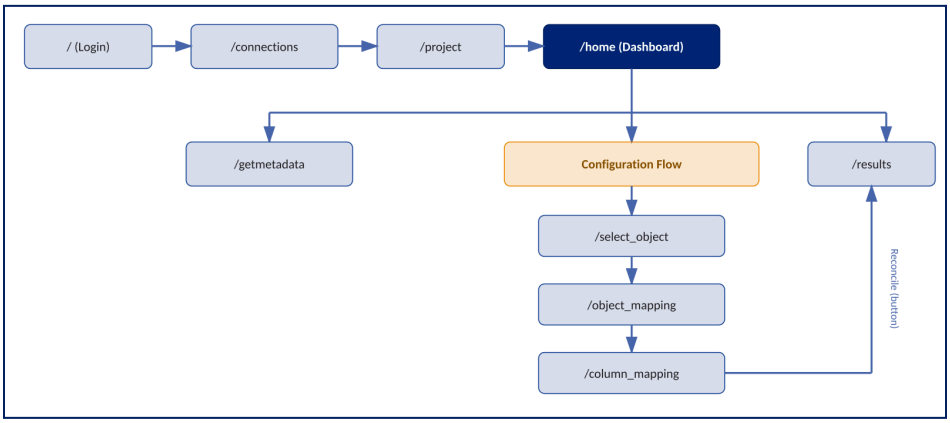

Fig 2 — Page Flow & Navigation through the Pi-Recon Web Application

5.2 UI Screens

5.2.1 Connection Management (/connections)

- Add/edit/delete database connections
- Supported types: MySQL, Snowflake, MSSQL
- Live connection testing before save
- Credentials encrypted before storage

5.2.2 Project Management (/project, /Add_new_project)

- Create reconciliation projects linking source connection to target connection
- Edit and delete projects

5.2.3 Metadata Extraction (/getmetadata)

- Background extraction of schemas, tables, and columns from source and target
- Progress polling with status updates
- Stores results in DISCOVERY.METADATA

5.2.4 Object Selection & Table Mapping (/select_object, /object_mapping)

- Select which tables from source to include in reconciliation
- Map source tables to target tables (1:1 mapping)

5.2.5 Column Mapping & Validation Rules (/column_mapping)

- Map source columns to target columns
- Configure per-column: key column flag, ignore flag, aggregation types (sum, mean, median, count)
- Supports composite key columns

5.2.6 Reconciliation Execution (/home → Reconcile button)

- One-click execution from the dashboard
- Connects to source and target, pulls data, compares
- Statistical sampling applied when configured
- Results written to RESULT schema

5.2.7 Results Dashboard (/results)

- Run history with match percentages
- Drill-down by database, schema, table, column
- Row-level mismatch details with source vs target values
- Column-level variance reports
- Excel export capability

6. Technical Details – Backend

The secure, versioned instance of the Pi-Recon app installed in the consumer account. It encapsulates all internal schemas used to store project configurations, connection credentials, metadata discovery, and reconciliation results.

6.1 Database Details

6.1.1 Database Details in Snowflake Environment

Sr. No	Object Name	Object Type	Description
1	PI_RECON_APP	APPLICATION_DATABASE	The secure, versioned instance of Pi-Recon installed in the consumer account. Encapsulates all schemas for project management, metadata discovery, and reconciliation results.

6.2 Schema Details

6.2.1 Schema Details in Snowflake Environment

Sr. No	Layer	Object Name	Object Type	Description
1	Core	CORE	SCHEMA	Application logic schema containing stored procedures, callbacks, and service management.
2	Configuration	PROJECT_DETAIL	SCHEMA	Stores project definitions, encrypted connection credentials, user accounts, and access control
3	Discovery	DISCOVERY	SCHEMA	Stores extracted metadata from source/target databases, object selections, table mappings, and column validation rules
4	Results	RESULT	SCHEMA	Stores reconciliation run history, row-level results, column-level results
5	Temporary	DATA	SCHEMA	Working schema for temporary data during reconciliation execution

6.3 Table Details

6.3.1 Tables in PROJECT_DETAIL Schema

Sr. No	Object Name	Object Type	Description
1	PROJECT_DETAIL.PROJECT	TABLE	Project definitions linking source and target systems with metadata
2	PROJECT_DETAIL.CONNECTIONS	TABLE	Encrypted connection credentials for MySQL, MSSQL, and Snowflake systems
3	PROJECT_DETAIL.PROJECT_ACCESS	TABLE	User-level information about project creation

6.3.2 Tables in DISCOVERY Schema

Sr. No	Object Name	Object Type	Description
1	DISCOVERY.METADATA	TABLE	Extracted schema, table, and column metadata from connected databases
2	DISCOVERY.OBJECT_SELECTION	TABLE	User-selected tables for reconciliation
3	DISCOVERY.OBJECT_MAPPING	TABLE	Source-to-target table mapping configuration
4	DISCOVERY.CONFIG	TABLE	Column-level mapping and validation rules (key columns, aggregations, ignore flags)

6.3.3 Tables in RESULT Schema

Sr. No	Object Name	Object Type	Description
1	RESULT.RUN_LOG	TABLE	Reconciliation execution history with status, duration, and record counts
2	RESULT.ROW_LEVEL_RESULT	TABLE	Row-by-row comparison results with source/target values and match status
3	RESULT.COLUMN_LEVEL_RESULT	TABLE	Aggregate comparison results with variance metrics

6.4 Object Details

6.4.1 Stored Procedures in Snowflake Environment

Sr. No	Object Name	Object Type	Description
1	CORE.GET_CONFIG_FOR_REFERENCE	PROCEDURE	Configuration callback that provides EAI configuration to Snowsight during reference binding
2	CORE.REGISTER_EAI_CALLBACK	PROCEDURE	Reference binding callback using SYSTEM$SET_REFERENCE for External Access Integration
3	CORE.REGISTER_WAREHOUSE_CALLBACK	PROCEDURE	Reference binding callback for the warehouse; automatically creates the compute pool and Pi-Recon service after warehouse binding
4	CORE.RESUME_SERVICE	PROCEDURE	Resumes the Pi-Recon SPCS service
5	CORE.CREATE_SRV	PROCEDURE	Manual fallback procedure to create the INFRA.PI_RECON_SERVICE service
6	CORE.SUSPEND_SERVICE	PROCEDURE	Suspends the Pi-Recon SPCS service to save compute costs
7	CORE.DROP_SERVICE	PROCEDURE	Drops the Pi-Recon SPCS service entirely
8	CORE.CREATE_COMPUTE_POOL	PROCEDURE	Manual fallback procedure to create the application compute pool

6.5 Snowflake Container Services (SPCS) Deployment

This section describes the deployment of the application using Snowpark Container Services within the Snowflake Native App Framework.

- **Packaging:** The reconciliation logic and web UI are containerized using Docker and bundled into the Application Package
- **Image Security:** Docker images are stored in an internal Provider Image Repository, scanned by Snowflake for security vulnerabilities prior to distribution
- **Service Instantiation:** A Snowflake Service is created within the Application Instance to run the containerized Flask web application
- **Isolated Compute:** The service executes reconciliation logic within an isolated, Snowflake-managed environment

- **Public Endpoint:** The service exposes a web UI endpoint accessible via Snowsight's "Launch App" button
- **Reference-Based Configuration:** Encryption keys, EAI, and warehouse are managed through manifest references and service specifications

6.5.1 Compute Pool Configuration

Sr. No	Object Name	Object Type	Description
1	<APP_NAME>_COMPUTE_POOL	COMPUTE POOL	Account-level compute resource provisioned automatically by the grant_callback. CPU_X64_XS instance family with auto-suspend at 600 seconds.

6.5.2 Snowflake Service Details

Sr. No	Service Name	Description
1	CORE.PI_RECON_SERVICE	Flask web application serving the reconciliation UI and executing comparison jobs within the SPCS container

6.5.3 Service Specification

The service is defined in service_spec.yaml:

- **Container Image:** /PI_SNOW/PI_RECON/PI_RECON_REPOSITORY/pi_recon_img:v1
- **Resources:** 512MB–2GB memory, 0.5–2 CPU cores
- **Secrets Injection:** SECRET_KEY and CONNECTION_ENCRYPTION_KEY from Snowflake Secret
- **Endpoint:** Port 9000, public, named "ui"
- **Warehouse:** Bound via REFERENCE('warehouse_ref') for query execution

6.6 Manifest Configuration

6.6.1 Privileges

Privilege	Description
CREATE COMPUTE POOL	Required to create compute pool for running the SPCS container
BIND SERVICE ENDPOINT	Required to expose the web UI endpoint for user access

6.6.2 References

Reference	Object Type	Required at Setup	Description
recon_secret_ref	SECRET	Yes	Encryption keys for session signing and credential encryption
recon_eai_ref	EXTERNAL ACCESS INTEGRATION	Yes	Network egress for external database connectivity
warehouse_ref	WAREHOUSE	Yes	Query warehouse for reconciliation execution

6.6.3 Grant Callback

The grant_callback property in the manifest specifies CORE.GRANT_CALLBACK, which is automatically invoked after the consumer grants privileges. It:

- Checks for CREATE COMPUTE POOL privilege → creates <APP_NAME>_COMPUTE_POOL
- Checks for BIND SERVICE ENDPOINT privilege → creates CORE.PI_RECON_SERVICE from specification file with EAI and warehouse references

7. Reconciliation Engine Details

7.1 Row-Level Reconciliation

- Connects to source and target databases using stored credentials
- Queries data based on configured table/column mappings
- Joins source and target datasets on designated key columns
- Compares each non-key column value pair
- Classifies results as MATCH or MISMATCH
- Records source value, target value, and record identifiers

7.2 Column-Level Reconciliation

- Executes aggregate functions (SUM, COUNT, MEAN, MEDIAN, AVERAGE) on configured columns
- Compares source aggregate against target aggregate
- Calculates variance between the two values
- Classifies as MATCH (variance = 0) or MISMATCH (variance != 0)

7.3 Sampling Methodology

- **Engine:** scipy.stats based calculation
- **Parameters:** Confidence Level (Z-score), Error Margin, Population Size, p/q proportions
- Formula: $n = (Z^2 \times p \times q) / E^2$ (adjusted for finite population)
- **Application:** Random sample of n records pulled from source/target for comparison
- **Override:** Consumer can disable sampling to compare all records

7.4 Key Column Matching

- One or more columns can be designated as "key columns" in column mapping
- Composite key support: multiple key columns are concatenated for matching
- Records are joined on key columns to establish source-target pairs
- Unmatched keys result in SOURCE_ONLY or TARGET_ONLY classifications

7.5 Result Classification

Result	Description
MATCH	All compared column values are identical between source and target
MISMATCH	One or more column values differ between source and target
SOURCE_ONLY	Record exists in source but not in target (based on key)
TARGET_ONLY	Record exists in target but not in source (based on key)

8. Troubleshooting Scenarios & Recommendations

Scenario	Description	Possible Causes	Recommended Actions
Connection Timeout	App fails to connect to external database	EAI not bound; Network rule missing host; Port blocked	Verify EAI binding in Settings; Add host:port to network rule via ALTER NETWORK RULE
Service Not Starting	Compute pool or service fails to create	CREATE COMPUTE POOL privilege not granted; Image pull failure	Re-grant privileges; Check Snowsight Monitoring for container errors
Metadata Extraction Fails	No tables/columns discovered	Credentials incorrect; Schema permissions insufficient	Test connection from Connections page; Verify user has SELECT on INFORMATION_SCHEMA
Reconciliation OOM	Service crashes during large comparison	Dataset too large for container memory	Enable statistical sampling; Reduce table scope
MFA Authentication Failure	Snowflake connection hangs/times out	Target account has MFA enforced with password auth	Switch to key-pair authentication for the target connection
Web UI Inaccessible	"App launch" button shows error	BIND SERVICE ENDPOINT not granted; Service suspended	Grant privilege; Call CORE.RESUME_SERVICE()

9. Glossary

Term	Definition
Application Instance	The deployed instance of the Pi-Recon app within the consumer's Snowflake account
Consumer Account	The Snowflake account where Pi-Recon is installed, configured, and executed
EAI	External Access Integration — a Snowflake security object enabling secure outbound connectivity
SPCS	Snowpark Container Services — Snowflake's managed container runtime for running Docker workloads
Reconciliation	The process of comparing data between two systems to identify discrepancies
Key Column	A column (or set of columns) used to uniquely identify records for matching
Variance	The numerical difference between source and target aggregate values
Fernet	A symmetric encryption algorithm from the cryptography library used for credential encryption
Reference Binding	Snowflake mechanism for securely associating consumer-owned objects (secrets, EAI) to a Native App
Grant Callback	A stored procedure automatically invoked by Snowflake after consumers grant privileges
Statistical Sampling	Mathematical technique to select a representative subset of data for comparison

10. Reference Artifacts & Editable Links

Asset Type	Description	Official Doc Link
Snowflake Platform Documentation	Official Snowflake documentation portal	docs.snowflake.com
Snowflake Container Services (SPCS)	Overview for running containerized workloads	Overview →
Snowflake Services	Creating, managing, and operating services	Services →
Compute Pools	Configuring compute pools for containers	Compute Pools →
Image Repository	Storing Docker images in Snowflake	Image Repository →
External Access Integration	Enabling secure outbound network access	EAI Reference →
Network Rules	Defining egress network rules	Network Rules →
Snowflake Secrets	Secure credential storage	Secrets Reference →
Native App Framework	Building and distributing Native Apps	Native Apps →
Native App Manifest	Manifest file reference	Manifest Reference →

11. Appendix

11.1 Usage Guidelines for Contributors

- Follow established naming conventions for Snowflake objects, application roles, and manifest-defined references
- All container images must undergo a mandatory security scan before publishing a new version
- Do not modify setup.sql or manifest.yml without an approved change request
- Ensure no credentials, tokens, or encryption keys are hardcoded in source code
- Test all changes against a local Docker environment before pushing to the image repository

11.2 Environment Deployment Guide

11.2.1 Prerequisites

- **Snowflake Account:** Enterprise Edition (or higher) with SPCS enabled in the target region
- **Administrative Privileges:** ACCOUNTADMIN or ability to grant CREATE COMPUTE POOL and BIND SERVICE ENDPOINT
- **Security Objects:** Encryption key secret (GENERIC_STRING) created in consumer account
- **Network Governance:** Network rules and EAI configured for external database egress
- **Warehouse:** An X-Small or Small warehouse available for query execution

11.2.2 Deployment Steps (Provider)

1. Build Docker image: docker build --platform linux/amd64 -t pi_recon_img:v1 .
2. Tag and push to Snowflake image repository
3. Upload manifest.yml, setup.sql, service_spec.yaml, README.md to named stage
4. Create Application Version: ALTER APPLICATION PACKAGE ... ADD VERSION V1_0 USING @stage
5. Create Marketplace listing and submit for security review

11.2.3 Installation Steps (Consumer)

1. Install from Marketplace — choose app name (e.g., PI_RECON_APP)
2. Create encryption key secret with secret_key and connection_encryption_key
3. Create network rule + EAI for external database hosts
4. Bind EAI, and Warehouse in Snowsight configuration screen
5. Grant CREATE COMPUTE POOL and BIND SERVICE ENDPOINT privileges
6. Wait for grant_callback to auto-create compute pool and service
7. Click "Open" to access Pi-Recon web UI
8. Update network rule with specific MySQL/MSSQL hosts as needed

11.2.4 Environment Configuration

- **Service Isolation:** Single SPCS service handles all reconciliation workloads

- **Credential Management:** All external DB credentials encrypted with consumer-owned Fernet key
- **Role Mapping:** Consumer maps APP_ADMIN to their Snowflake RBAC structure
- **Resource Sizing:** CPU_X64_XS compute pool; 512MB–2GB container memory

11.2.5 Validation

Once installation is complete, verify:

- **Reference Verification:** All references (Secret, EAI, Warehouse) show status 'ASSOCIATED'
- **Service Status:** SPCS service status is RUNNING (check in Snowsight > Monitoring)
- **UI Access:** Click "Open" on the app — Pi-Recon login page loads
- **Connection Test:** Add a test connection and verify "Connection successful" response
- **End-to-End:** Create a project, extract metadata, run a reconciliation, and view results

12. Support & Contact Information

12.1 Technical Support

For issues related to reconciliation failures, connectivity troubleshooting, or bug reporting:

Email: Admin@Pibythree.com, Alliance@Pibythree.com

Note: As this is a Snowflake Native App, please check the RESULT.RUN_LOG table for error messages before reaching out, as this will help our team expedite your request.

12.2 Business & Sales Inquiries

For inquiries regarding the broader Pi-Snow roadmap, licensing for additional database connectors, or custom integration needs:

Contact: Alliance@pibythree.com

Website: www.pibythree.com

12.3 Documentation & Feedback

To access the latest version of this document or provide feedback:

Documentation Portal: Pi Accelerators by PibyThree