π - Ingest

PostgreSQL CDC Native App

Technical Architecture, Implementation & Deployment Documentation

Version	01
Author	PibyThree Developers

πby3

Table of Contents

1 Introduction 3

2 Key Considerations 4

2.1 Scope 4

2.2 Out of Scope 4

2.3 Assumptions 4

2.4 Dependencies 4

2.5 Risks 5

3 Business Feature Details 6

3.1 Business Process Flow 6

3.2 RBAC & UBAC 6

3.3 Execution & Refresh Behaviour 6

3.4 Monitoring & Operational Visibility 6

4 Technical Feature Details 7

4.1 Technical Architecture 7

4.2 Transaction Processing & Data Integrity 7

4.3 Security & Secure Access Management 7

4.4 Configuration Management 8

5 Technical Details — Backend 9

5.1 Database Details 9

5.2 Schema Details 9

5.3 Table Details 9

5.4 Objects Details 11

5.5 SPCS Deployment 11

6 Troubleshooting Scenarios & Recommendations 13

7 Glossary 14

8 Reference Artifacts & Editable Links 15

9 Appendix 16

9.1 Usage Guidelines 16

9.2 Environment Deployment Guide 16

Document note: This document is based on the PostgreSQL CDC Native App README, manifest, and setup SQL available on 03-Nov-2026. Live Snowflake/PostgreSQL integration validation remains an environment-specific activity.

1 Introduction

This document describes the requirements, architecture, configuration, and implementation of the PostgreSQL Change Data Capture (CDC) Native App. The solution captures committed data changes from PostgreSQL through logical replication and applies them to Snowflake using a long-running Snowpark Container Services (SPCS) worker distributed through the Snowflake Native App framework.

The objective is to provide a secure, recoverable, and observable ingestion path for INSERT, UPDATE, and DELETE operations while preserving source transaction identity, durable checkpoints, and raw event history. The worker uses PostgreSQL's **pgoutput** logical decoding protocol and does not parse physical WAL files directly.

The application is intentionally headless. Pipeline configuration, schema migrations, monitoring, troubleshooting, and recovery are performed through Snowflake SQL and version-controlled Python, YAML, and SQL assets. No custom Streamlit or dashboard UI is included in this release.

The current reliability release adds durable transaction-parent and sub-batch tracking, deterministic event hashes, retry classification, quarantine, replication-slot/WAL monitoring, schema versioning, reconciliation, and a guarded process for slot acknowledgement positions that are ahead of the Snowflake checkpoint.

This document is intended for solution architects, PostgreSQL administrators, Snowflake administrators, data engineers, testers, and operational support teams.

2 Key Considerations

This section outlines the boundaries, assumptions, dependencies, and operational risks that influence implementation and deployment.

2.1 Scope

- Capture PostgreSQL row changes through logical replication using a publication, replication slot, and pgoutput.
- Run the CDC consumer and Snowflake applier inside a Snowpark Container Services worker.
- Archive source change events in the application database under PG.PG_RAW.
- Apply the same staged sub-batch directly to mapped Snowflake target tables in PG_INGEST.
- Persist source checkpoints, transaction/sub-batch status, event hashes, audit details, and error/quarantine records.
- Support controlled additive schema evolution, configuration revisioning, retries, WAL monitoring, and reconciliation.

2.2 Out of Scope

- Physical WAL file parsing or direct interpretation of PostgreSQL WAL segments.
- Automatic initial snapshot or historical backfill of existing source rows.
- Automatic dropping or renaming of source/target tables or columns.
- Automatic alteration of existing target column types, precision, scale, or nullability.
- Multiple concurrent consumers sharing one logical replication slot.
- An in-app dashboard, Streamlit UI, or end-user visual configuration interface.
- Full row-by-row reconciliation of every PostgreSQL source table against its Snowflake target.

2.3 Assumptions

Sr. No.	Description	Implications, if not met
1	PostgreSQL supports logical replication and pgoutput.	CDC stream cannot be established.
2	The publication includes every configured source table and required DML operations.	Changes outside publication membership are not delivered.
3	The replication slot is available and used by one worker only.	Concurrent use or slot loss can compromise continuity.
4	Source tables have a primary key or suitable non-null unique key with compatible replica identity.	UPDATE/DELETE target application cannot be safely keyed.
5	The consumer account has SPCS, image repository, EAI, secret, warehouse, and required grants.	The Native App service cannot be provisioned or connect.
6	A consistent initial load is performed separately when source tables already contain data.	Targets otherwise contain only changes consumed after CDC starts.

2.4 Dependencies

Sr. No.	Dependency	Owner	Implications, if not met
1	PostgreSQL logical replication settings, publication and slot	PostgreSQL DBA	No logical change stream.
2	PostgreSQL credentials with login and replication privileges	PostgreSQL DBA	Worker cannot authenticate or start replication.
3	Snowflake Native App package and image repository	Provider / App team	Application version or image cannot be deployed.

Sr. No.	Dependency	Owner	Implications, if not met
4	PASSWORD secret, EAI, warehouse and compute pool	Consumer Snowflake Admin	Service cannot access PostgreSQL or Snowflake.
5	Docker Desktop and Snowflake CLI	Deployment Engineer	Image build, push, and app deployment cannot complete.
6	Approved SQL configuration and version increment	Data Engineering	Worker may continue with stale or inconsistent mappings.

2.5 Risks

Sr. No.	Risk	Mitigation Strategy
1	Replication slot retains WAL while the consumer is stopped.	Monitor retained WAL; resolve failures promptly; never drop the slot as a routine cleanup.
2	Checkpoint advances before all target/archive writes are durable.	Use transaction and sub-batch ledgers; acknowledge only the last durable Snowflake checkpoint.
3	Duplicate replay after restart or uncertain commit.	Validate transaction/chunk hashes and completed sub-batch records before skipping replay.
4	Schema drift causes incompatible target writes.	Auto-add compatible columns; quarantine and block incompatible changes for explicit migration.
5	Configuration changes are only partially applied.	Update config rows and CONFIG_VERSION in the same SQL transaction.
6	Slot confirmed_flush_lsn is ahead of Snowflake checkpoint.	Record exact LSN pair and require review/approval or repair before resuming.

3 Business Feature Details

This section describes the operational behaviour from the perspective of data consumers and platform administrators.

3.1 Business Process Flow

1. PostgreSQL commits a source transaction and makes its logical changes available through the configured publication and replication slot.
2. The SPCS worker reads pgoutput messages and assembles committed source transactions.
3. Events are staged in a session-scoped Snowflake temporary table and divided into configured sub-batches when required.
4. Each sub-batch is written to the raw archive and applied to mapped target tables within the same Snowflake transaction.
5. Durable child sub-batch and parent transaction ledgers are updated. The parent is complete only when every expected child is complete.
6. The worker advances the Snowflake checkpoint and sends PostgreSQL feedback only through the durable checkpoint.

3.2 RBAC & UBAC

- The Native App defines the APP_ADMIN application role for controlled access to internal schemas, tables, and management procedures.
- Consumer account administrators bind the required references and grant the account-level CREATE COMPUTE POOL privilege to the installed application.
- Application objects are accessed through the application role and its granted privileges; consumer administrators retain responsibility for account-level reference binding and grants.
- Target data is stored in the application database and should be granted only to authorized consumer roles according to the organization's data governance policy.
- Snowflake standard-table primary keys are informational and are not enforced; operational uniqueness is maintained by the worker's single-consumer design and controlled configuration.

3.3 Execution & Refresh Behaviour

The service is designed as a continuously running CDC consumer rather than a scheduled full-refresh job. The configured batch size and batch timing govern how incoming changes are grouped for processing. Large source transactions can be split into sequential committed chunks using PG_APPLY_SUB_BATCH_SIZE. Each chunk is atomic, but earlier chunks may become visible to target readers before the full source transaction is complete. The source checkpoint remains unchanged until all chunks finish successfully.

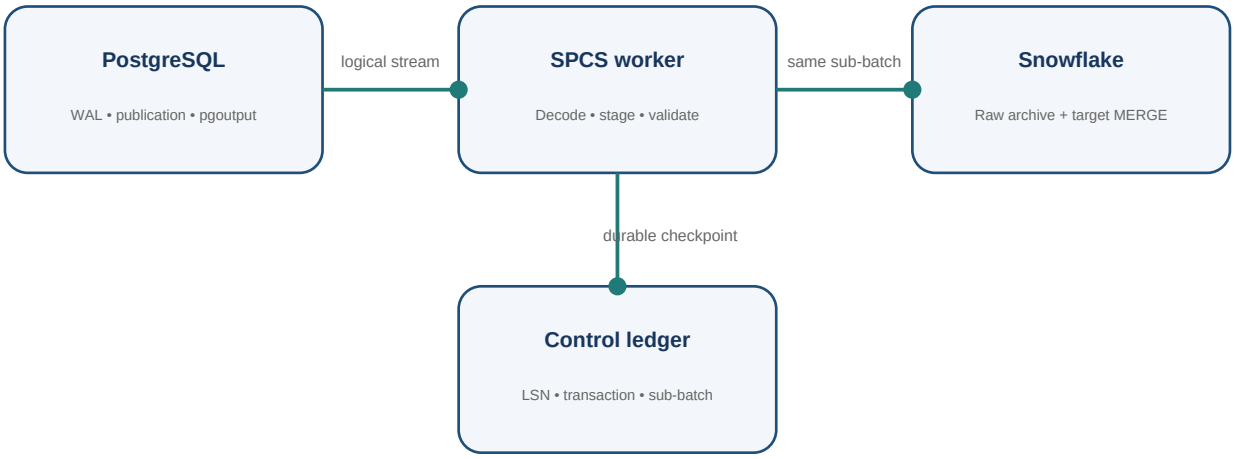
3.4 Monitoring & Operational Visibility

- Pipeline status and last processed LSN are available from CONFIG_PG.GET_PG_PIPELINE_STATUS and PG.PG_CONTROL.
- PG.PG_SLOT_MONITOR captures slot activity, plugin, restart LSN, confirmed flush LSN, current WAL LSN, and retained/unconfirmed WAL bytes.
- LOGS_PG.AUDIT_LOG and LOGS_PG.ERROR_LOG capture operational execution status and failures.
- PG.PG_BATCH_AUDIT, PG.CDC_TRANSACTION_CONTROL, and PG.CDC_SUB_BATCH_CONTROL expose batch and transaction completion.
- PG.PG_SCHEMA_CHANGE_AUDIT, PG.PG_RECONCILIATION_AUDIT, PG.CDC_QUARANTINE, and PG.PG_SLOT_ACK_INCIDENT support controlled investigation and recovery.

4 Technical Feature Details

This section describes the core architecture, transaction semantics, security model, and configuration lifecycle.

4.1 Technical Architecture



The PostgreSQL logical replication stream is consumed by a Python worker running in SPCS. The worker stages the incoming events in a session-scoped temporary table. From that same stage, it writes the raw event archive and applies changes directly to the configured Snowflake target tables. The target apply path does not replay data by reading PG.PG_RAW.

4.2 Transaction Processing & Data Integrity

- **Atomic sub-batch:** raw archive, target changes, and durable child status are committed together for each sub-batch.
- **Parent transaction ledger:** PG.CDC_TRANSACTION_CONTROL stores source commit LSN, XID, event count, transaction hash, expected/completed child count, status, and error details.
- **Child ledger:** PG.CDC_SUB_BATCH_CONTROL stores sub-batch identity, deterministic event hash, event count, chunk plan, and completion state.
- **Idempotent replay:** replayed work is checked against parent transaction identity, chunk plan, event count, and payload hash before completed work is skipped.
- **Checkpoint discipline:** PostgreSQL feedback is limited to the last durable Snowflake checkpoint. The checkpoint is not advanced until all chunks of the source transaction complete.

Large transaction behaviour

A source transaction larger than PG_APPLY_SUB_BATCH_SIZE is processed as sequential, independently committed chunks. This reduces the size of individual Snowflake transactions. It does not provide transaction-wide visibility across all chunks: target readers may observe earlier chunks before later chunks are committed. Set a larger chunk size when transaction-wide visibility is important, while respecting Snowflake statement and transaction limits.

4.3 Security & Secure Access Management

- **PostgreSQL credentials:** stored in a consumer-owned Snowflake PASSWORD secret and mounted into the container through pg_credentials_ref. No password is embedded in Python, Dockerfile, manifest, or service specification.
- **External connectivity:** pg_external_access_ref binds a consumer-managed External Access Integration. Its network rule must allow the exact PostgreSQL endpoint and port 5432.
- **Warehouse:** warehouse_ref binds the warehouse used by the worker for Snowflake SQL execution.

- **Compute:** the application requires CREATE COMPUTE POOL on the consumer account to provision the SPCS compute pool.
- **Least privilege:** the APP_ADMIN application role receives explicit access to application schemas, tables, procedures, compute pool, and service operations.

4.4 Configuration Management

CONFIG_PG is the source of truth for connection, publication, slot, pipeline batch size, table mappings, key columns, and delete mode. The service specification deliberately avoids overriding these settings with environment variables. The worker polls CONFIG_PG.CONFIG_VERSION (30 seconds by default) and reloads validated mappings between source transactions.

Configuration change control

Every change to PIPELINE_CONFIG, CONNECTION_CONFIG, TABLE_CONFIG, or COLUMN_CONFIG must increment CONFIG_VERSION in the same SQL transaction. Use sql/config_change_template.sql. Changes to host, port, database, credentials, publication, or slot require a controlled service restart; they are not accepted as a live mapping reload. A configuration-table edit alone does not update the EAI network allowlist.

5 Technical Details — Backend

This section lists the application database objects and their purpose.

5.1 Database Details

The Native App creates its internal schemas and objects in the installed application database. The sample deployment uses POSTGRES_CDC_NATIVE_APP as the app database name; object names should be qualified with the actual installed application database in the consumer account.

5.1.1 Database Details in Snowflake Environment

Sr. No.	Database / Context	Description
1	POSTGRES_CDC_NATIVE_APP	Sample installed application database containing configuration, control, logs, raw archive, and target schemas.
2	Consumer PostgreSQL database	Source database configured in CONFIG_PG.CONNECTION_CONFIG; the setup seed intentionally leaves consumer-specific values blank.
3	Snowflake warehouse	Bound through warehouse_ref and used for worker SQL execution.

5.2 Schema Details

5.2.1 Schema Details in Snowflake Environment

Sr. No.	Schema	Purpose
1	CONFIG_PG	Pipeline and connection settings, table/column mappings, config version, reference callbacks, and service procedures.
2	PG	Raw CDC archive, checkpoint, batch/transaction ledgers, schema metadata, slot telemetry, reconciliation, and quarantine.
3	PG_INGEST	Target tables populated from PostgreSQL row changes.
4	LOGS_PG	Operational audit and error logs.

5.3 Table Details

5.3.1 Tables Details in Snowflake Environment

Layer	Object Name	Description
Configuration	CONFIG_PG.PIPELINE_CONFIG	Pipeline name, connection, publication, slot, batch size, batch timing, and active flag.
Configuration	CONFIG_PG.CONNECTION_CONFIG	PostgreSQL host, port, database, username metadata, secret reference name, SSL mode, and active flag.
Configuration	CONFIG_PG.TABLE_CONFIG	Source-to-target table mapping, key columns, delete mode, soft-delete column, and active status.
Configuration	CONFIG_PG.COLUMN_CONFIG	Source-to-target column mapping, key indicator, ordinal position, and active status.
Configuration	CONFIG_PG.CONFIG_VERSION	Monotonic configuration revision used by the worker for safe reloads.
Control	PG.PG_CONTROL	Durable slot/publication checkpoint and last processed LSN/batch identity.
Raw	PG.PG_RAW	Append-oriented archive of operation, before/after payload, LSN, transaction ID, commit timestamp, and event sequence.
Audit	PG.PG_BATCH_AUDIT	Batch end LSN, event count, status, and error message.

Layer	Object Name	Description
Control	PG.CDC_TRANSACTION_CONTROL	Parent ledger for source transaction completion and expected/completed sub-batches.
Control	PG.CDC_SUB_BATCH_CONTROL	Child ledger with deterministic event hash and sub-batch completion state.
Control	PG.CDC_QUARANTINE	Permanent data/config/schema failures, payload evidence, classification, attempts, and retry state.
Monitoring	PG.PG_SLOT_MONITOR	Replication slot and WAL retention telemetry.
Monitoring	PG.PG_RECONCILIATION_AUDIT	Results of durable transaction/archive consistency checks.
Schema	PG.PG_SCHEMA_METADATA	Source/target schema fingerprint, columns, key, replica identity, and schema version.
Schema	PG.PG_SCHEMA_CHANGE_AUDIT	Detected schema changes and migration review status.
Safety	PG.PG_SLOT_ACK_INCIDENT	Exact slot confirmed-flush and Snowflake checkpoint LSN pair requiring review under the release's recovery policy.
Audit	PG.PG_APPLY_CONTROL / PG.PG_APPLY_AUDIT	Per-table apply progress and apply execution statistics.
Logs	LOGS_PG.AUDIT_LOG / LOGS_PG.ERROR_LOG	Execution history and detailed errors.

5 Technical Details — Backend (Continued)

5.3.2 Starter Target Tables

Target Table	Representative Columns	Key / Behaviour
PG_INGEST.CUSTOMERS	ID, NAME, CITY, IS_DELETED	ID is configured as the key; soft delete is the default for newly seeded mappings.
PG_INGEST.EMPLOYEE	EMPLOYEE_ID, EMPLOYEE_NAME, DEPARTMENT, SALARY, CITY, EMAIL, SYSMODTIMESTAMP, IS_DELETED	EMPLOYEE_ID is configured as the key; SYSMODTIMESTAMP is a source-managed update timestamp.

Delete behaviour

New mappings default to SOFT delete. A PostgreSQL DELETE marks the Snowflake-only IS_DELETED column TRUE while retaining the target row. An INSERT or UPDATE sets IS_DELETED to FALSE, allowing a retained key to become active again. HARD delete remains available when a table is intentionally configured for physical deletion. Existing mappings are consumer-managed and should be migrated explicitly.

5.4 Objects Details

5.4.1 Object Details in Snowflake Environment

Sr. No.	Object Name	Object Type	Description
1	CONFIG_PG.GET_CONFIG_FOR_REFERENCE	Procedure	Returns reference configuration for supported Native App references.
2	CONFIG_PG.REGISTER_REFERENCE_CALLBACK	Procedure	Registers, removes, or clears application references.
3	CONFIG_PG.CREATE_PG_SERVICE	Procedure	Creates the compute pool and SPCS service using the service specification template.
4	CONFIG_PG.RESUME_PG_SERVICE	Procedure	Resumes the CDC service.
5	CONFIG_PG.SUSPEND_PG_SERVICE	Procedure	Suspends the CDC service.
6	CONFIG_PG.DROP_PG_SERVICE	Procedure	Drops the CDC service when required for controlled recreation.
7	CONFIG_PG.GET_PG_SERVICE_STATUS	Procedure	Returns service status information.
8	CONFIG_PG.GET_PG_PIPELINE_STATUS	Procedure	Returns pipeline configuration and latest checkpoint status.

5.5 Snowflake Container Services (SPCS) Deployment

The Python worker is packaged as a Linux container image and deployed as a long-running SPCS service inside the Native App installation. The image is stored in a Snowflake Image Repository and referenced by the application manifest and service specification.

5.5.1 Compute Pool Configuration

Parameter	Current documented value / approach
Compute pool	PG_COMPUTE_POOL (created by the app procedure)
Node range	MIN_NODES = 1; MAX_NODES = 1
Instance family	CPU_X64_XS

Parameter	Current documented value / approach
Auto resume / suspend	AUTO_RESUME = TRUE; AUTO_SUSPEND_SECS = 60
Application privilege	CREATE COMPUTE POOL on account, granted to installed application.

5.5.2 Snowflake Service Details

Parameter	Value / Description
Service name	PG_CDC_SERVICE under CONFIG_PG (derived fully qualified name depends on installed app database).
Container image	pg-cdc-consumer:1.14
Native App version	V1_0_7
External access	pg_external_access_ref bound to an EAI allowing the PostgreSQL endpoint.
Warehouse	warehouse_ref bound to the consumer warehouse.
Credential mount	pg_credentials_ref PASSWORD secret mounted into the worker container.
Scaling	Single instance / single consumer per replication slot.

6 Troubleshooting Scenarios & Recommendations

Use the diagnostics and recovery controls before changing checkpoints, slots, or durable control tables.

Scenario	Description / Possible Causes	Recommended Actions
Service does not start	Unbound secret/EAI/warehouse reference; missing CREATE COMPUTE POOL; invalid image path/tag; insufficient application privileges.	Verify all references are ASSOCIATED; check image tag consistency; confirm account-level grant and service status.
PostgreSQL connection failure	Incorrect host/port/database; expired password; EAI network rule does not permit the endpoint; PostgreSQL is not reachable.	Validate CONFIG_PG.CONNECTION_CONFIG, rotate/update the PASSWORD secret, confirm EAI network rule and PostgreSQL firewall.
No new rows are captured	Publication omits the table or DML operation; worker is suspended; table has no valid key; no new DML occurred.	Check publication membership, service status, slot activity, table key/replica identity, and latest checkpoint.
WAL retention is increasing	Worker stopped or is not acknowledging durable progress; long-running or blocked transactions.	Inspect PG.PG_SLOT_MONITOR and ERROR_LOG; resolve the worker issue. Never drop the slot to clear WAL.
Schema drift / apply blocked	Source type, length, precision, scale, nullability, drop, or rename differs from the target metadata.	Inspect PG.PG_SCHEMA_CHANGE_AUDIT and PG.PG_SCHEMA_METADATA. Apply an explicit migration, update mappings, increment CONFIG_VERSION, and retry.
Quarantine record created	Permanent data/config/schema failure or non-transient apply error.	Inspect PG.CDC_QUARANTINE and PG.PG_BATCH_AUDIT; fix root cause; then mark RETRY_REQUESTED using sql/operations.sql.
Slot acknowledgement ahead	PostgreSQL confirmed_flush_lsn is ahead of the durable Snowflake checkpoint.	Inspect exact LSN pair in PG.PG_SLOT_ACK_INCIDENT; reconcile source/target data and approve only the exact pair after review.
Configuration appears stale	A config table changed without CONFIG_VERSION increment or update occurred during an active source transaction.	Use sql/config_change_template.sql; update config and revision in one transaction; restart for connection/publication/slot changes.
Target table not created	CREATE TABLE DDL itself is not a row-level pgoutput event; no first relation/DML activity; table not in allowed schema or publication.	Ensure publication coverage and allowed schema; issue a row change and verify relation/DML event arrives.
Duplicate or inconsistent replay	Concurrent worker on same slot; manual edits to control tables; incomplete transaction/sub-batch history.	Keep one worker per slot; do not delete or truncate control/checkpoint ledgers; inspect hashes and transaction completion before recovery.

Important recovery rule: do not delete or truncate PG.PG_CONTROL, PG.CDC_TRANSACTION_CONTROL, PG.CDC_SUB_BATCH_CONTROL, or PG.CDC_QUARANTINE as a troubleshooting shortcut. The PostgreSQL slot is the replay source for unacknowledged changes.

7 Glossary

Term	Definition
CDC	Change Data Capture — identifying and transferring inserts, updates, and deletes from a source system.
LSN	Log Sequence Number — PostgreSQL position marker used to track logical replication progress.
WAL	Write-Ahead Log — PostgreSQL log stream from which logical decoding produces changes.
pgoutput	PostgreSQL built-in logical replication output plugin used by the consumer.
Publication	PostgreSQL object that defines which tables and DML changes are exposed to logical replication.
Replication slot	PostgreSQL object that retains required WAL until the consumer acknowledges progress.
SPCS	Snowpark Container Services — Snowflake-managed environment for running containerized workloads.
Native App	Snowflake application distribution model that packages application code, objects, and privileges.
EAI	External Access Integration — Snowflake object controlling permitted outbound network access.
PASSWORD secret	Snowflake secret object that securely stores username/password credentials.
Sub-batch	A sequential chunk of events from one source transaction, committed independently in Snowflake.
Idempotency	Property that allows replay without applying an already completed event chunk more than once.
Quarantine	Durable record of a failed batch that requires investigation and controlled retry.
Schema fingerprint	Deterministic representation of source/target schema attributes used to detect schema changes.
CONFIG_VERSION	Monotonic configuration revision used to trigger safe worker mapping reloads.
Soft delete	Logical deletion that retains the Snowflake target row and sets IS_DELETED to TRUE.

8 Reference Artifacts & Editable Links

This section lists the implementation assets and official platform documentation used by the solution.

Asset Type	Description	Reference / Location
Worker source	CDC consumer, apply engine, retry, monitoring, schema management, and shutdown handling.	app/pg_cdc_worker.py
Native App setup	Application role, schemas, configuration/control tables, reference callbacks, and management procedures.	app/setup.sql
Manifest	Native App version and image declaration, privileges, and references.	app/manifest.yml
Service specification	Container image, secret mount, runtime settings, and service configuration.	app/service_spec.yaml
Build assets	Dockerfile, requirements, and .dockerignore.	Project root
Configuration migration	Atomic configuration update and CONFIG_VERSION increment template.	sql/config_change_template.sql
Operational SQL	Diagnostics, WAL/slot inspection, reconciliation, quarantine, and controlled retry.	sql/operations.sql
Upgrade script	Support DDL for existing app database installations when required.	sql/upgrade_v1_0_4.sql
Unit/static tests	Local tests for reliability helpers and slot acknowledgement recovery gate.	tests/test_hardening.py

Official Snowflake documentation

Topic	Documentation URL
Snowflake Native Apps	https://docs.snowflake.com/en/developer-guide/native-apps/about
Snowpark Container Services overview	https://docs.snowflake.com/en/developer-guide/snowpark-container-services/overview
SPCS services	https://docs.snowflake.com/en/developer-guide/snowpark-container-services/services
Compute pools	https://docs.snowflake.com/en/developer-guide/snowpark-container-services/compute-pools
Image repositories	https://docs.snowflake.com/en/developer-guide/snowpark-container-services/image-repository
External Access Integration	https://docs.snowflake.com/en/sql-reference/sql/create-external-access-integration
Network rules	https://docs.snowflake.com/en/sql-reference/sql/create-network-rule
Snowflake secrets	https://docs.snowflake.com/en/sql-reference/sql/create-secret
PostgreSQL logical replication	https://www.postgresql.org/docs/current/logical-replication.html
PostgreSQL logical decoding	https://www.postgresql.org/docs/current/logicaldecoding.html

9 Appendix

Supporting usage guidelines, deployment instructions, environment configuration, and validation checks.

9.1 Usage Guidelines for Contributors

- Follow established naming conventions for application schemas, procedures, references, and container images.
- Keep the manifest image tag and service specification image tag synchronized.
- Do not commit credentials, tokens, or private connection values to source control.
- All pipeline configuration edits must increment CONFIG_VERSION atomically with the configuration change.
- Keep exactly one active worker per logical replication slot.
- Do not manually alter checkpoint, transaction, sub-batch, quarantine, or slot-incident history without an approved recovery procedure.
- Test Python syntax, unit tests, YAML parsing, and SQL structure before packaging a new application version.
- Use an approved versioning plan when changing setup.sql, manifest.yml, service_spec.yaml, or persistent table definitions.

9.2 Environment Deployment Guide

This section provides the deployment sequence for installing or upgrading the PostgreSQL CDC Native App in a consumer Snowflake account.

9.2.1 Prerequisites

- PostgreSQL instance configured for logical replication (wal_level = logical), with sufficient replication slots and WAL senders.
- Publication supports the required INSERT, UPDATE, and DELETE operations and includes all configured source tables.
- Existing replication slot uses pgoutput and is dedicated to this worker.
- Snowflake Native App project root and snowflake.yml are available; the current project-specific snowflake.yml must be retained.
- Docker Desktop is installed and running; Snowflake CLI is authenticated and has image push / Native App deployment privileges.
- Consumer has a PASSWORD secret, EAI, warehouse, SPCS support, and an administrator able to grant CREATE COMPUTE POOL.

9.2.2 Deployment Steps

1. Review the current app files and ensure the manifest and service specification both reference the intended image tag. The documented V1_0_7 release uses pg-cdc-consumer:1.14.
2. Run app/setup.sql for a new installation. For an existing V1_0_6 installation, apply the V1_0_7 setup changes through the Native App upgrade path; use sql/upgrade_v1_0_4.sql only if the earlier V1_0_4 support objects/grants were never installed.
3. Configure CONFIG_PG.CONNECTION_CONFIG and CONFIG_PG.PIPELINE_CONFIG with the actual PostgreSQL host, port, database, connection name, publication, and slot. Keep the password in the bound PASSWORD secret.
4. Confirm the bound EAI network rule permits the actual PostgreSQL host/IP and TCP port 5432. Updating the connection table does not grant network egress.
5. Confirm publication membership and replication slot settings. Ensure only one worker uses the slot.
6. Run SHOW IMAGE REPOSITORIES IN SCHEMA POSTGRES_PG.PG; and copy the exact repository_url returned for PG_IMAGE_REPO.
7. Build and push the container image from the project directory using the following PowerShell commands.

```
$repo = "&lt;REPOSITORY_URL_FROM_SHOW_IMAGE_REPOSITORIES&gt;"
$image = "$repo/pg-cdc-consumer:1.14"
docker build --platform linux/amd64 -t $image .
snow spcs image-registry login --connection YOUR_SNOWFLAKE_CONNECTION
docker push $image
```

1 Deploy the Native App package/version using the existing project configuration.

```
snow app run --connection YOUR_SNOWFLAKE_CONNECTION
```

1 Update the existing SPCS service to the new specification and image. For the documented app database/service name:

```
ALTER SERVICE POSTGRES_CDC_NATIVE_APP.CONFIG_PG.PG_CDC_SERVICE
FROM SPECIFICATION_TEMPLATE_FILE = '/service_spec.yaml'
USING (app_database =&gt; '"POSTGRES_CDC_NATIVE_APP"');
```

1 Bind pg_credentials_ref to the consumer-owned PASSWORD secret, pg_external_access_ref to the EAI, and warehouse_ref to the required warehouse. Grant CREATE COMPUTE POOL on account to the installed application.

2 Resume the service and verify service status, pipeline status, checkpoint progress, raw archive, target tables, and audit/error logs.

9.2.3 Environment Configuration

Configuration	Required Value / Guidance
Connection	CONFIG_PG.CONNECTION_CONFIG is authoritative for host, port, database, username metadata, secret reference, SSL mode, and active status.
Pipeline	CONFIG_PG.PIPELINE_CONFIG defines pipeline name, connection name, publication, slot, batch size, batch seconds, and active status.
Table mapping	CONFIG_PG.TABLE_CONFIG defines source/target table names, key columns, delete mode, soft-delete column, and active status.
Column mapping	CONFIG_PG.COLUMN_CONFIG defines source/target column names, key flag, order, and active status.
References	Bind pg_credentials_ref, pg_external_access_ref, and warehouse_ref in the installed Native App.
Configuration revision	Increment CONFIG_PG.CONFIG_VERSION in the same transaction as every mapping/configuration update.

9.2.4 Validation

- Reference Verification: confirm the PostgreSQL PASSWORD secret, EAI, and warehouse references are bound and usable.
- Service Status: verify the SPCS service and container are running using the app status procedure and Snowflake service inspection commands.
- Pipeline Health: call CONFIG_PG.GET_PG_PIPELINE_STATUS and confirm the latest processed LSN advances after source DML.
- Raw Archive: inspect PG.PG_RAW for the expected operation, table, LSN, transaction ID, and event payload.
- Target Apply: verify matching INSERT/UPDATE/DELETE outcomes in PG_INGEST target tables.
- Transaction Integrity: confirm completed parent transactions have all expected sub-batches completed.
- Monitoring: review PG.PG_SLOT_MONITOR for slot activity and WAL retention; inspect reconciliation and quarantine tables for unresolved records.
- Security: verify credentials are absent from source files and the EAI permits only the required endpoint/port.

Validation status: local syntax/unit/static checks are described in the release README. Live SQL execution against the consumer Snowflake account and end-to-end replication against the PostgreSQL instance must be validated in the target environment before production use.

9 Appendix — Release Notes & Operational Limitations

9.3 Reliability and Data Integrity Release

Capability	Documented Behaviour
Configuration revisioning	CONFIG_VERSION polled every 30 seconds by default; safe mapping reload between source transactions.
Graceful shutdown	SIGTERM/SIGINT handling; finish queued committed work where possible; feedback constrained to durable checkpoint.
Retry policy	Transient errors use bounded exponential backoff with jitter; invalid credentials/privileges are fatal; permanent data/config/schema failures are quarantined.
Idempotency	SHA-256 event hashes and parent/child ledgers validate replay identity before completed work is skipped.
Slot monitoring	Default 60-second sampling; warning at 5 GiB retained WAL and critical at twice the threshold.
Reconciliation	Default 5-minute checks for transaction parent/child consistency, checkpoint state, recent raw archive counts, and unresolved quarantine.
Schema policy	Additive nullable columns supported; destructive or incompatible changes require explicit migration and review.
Quarantine	Unacknowledged source WAL is replayed after root-cause correction and controlled retry request.

9.4 Environment Defaults

Setting	Default	Purpose
PG_MAX_RETRY_ATTEMPTS	7	Maximum attempts for transient failures.
PG_RETRY_BASE_SECONDS	1	Initial exponential backoff interval.
PG_RETRY_MAX_SECONDS	30	Maximum backoff interval.
PG_SLOT_MONITOR_INTERVAL_SECONDS	60	Slot/WAL sampling cadence.
PG_RECONCILE_INTERVAL_SECONDS	300	Durable-state reconciliation cadence.
PG_MAX_RETAINED_WAL_BYTES	5368709120	WAL warning threshold (5 GiB).
PG_QUARANTINE_MAX_PAYLOAD_BYTES	8388608	Maximum forensic payload snapshot size (8 MiB).
PG_CONFIG_POLL_SECONDS	30	Configuration revision polling cadence.
PG_APPLY_SUB_BATCH_SIZE	10000	Maximum events per committed source-transaction sub-batch.

9.5 Known Limitations and Release Alignment

- No initial snapshot is created. Perform a consistent initial load separately before relying on CDC for a pre-populated source table.
- Automatic table discovery occurs when the worker first receives relation/DML activity for a published table; CREATE TABLE itself is not a row-level pgoutput event.
- Column/table drops and renames are not automated. Incompatible changes are blocked and require explicit SQL migration.
- Reconciliation validates durable pipeline ledgers and recent raw-archive counts; it is not a complete source-to-target row-by-row comparison.
- The documented application is single-consumer/single-slot. Do not deploy parallel workers against the same slot.

- The release README and manifest surfaced for this documentation identify V1_0_7 and image 1.14. Ensure the actual deployment manifest and service specification use the same release identifiers before publishing.

9.6 Support & Contact Information

For issues related to PostgreSQL connectivity, ingestion failures, SPCS deployment, or data-integrity recovery, contact the designated PibyThree project support team through the organization's approved support channel. Before escalating, capture the service status, relevant PG.PG_CONTROL checkpoint, PG.PG_BATCH_AUDIT row, LOGS_PG.ERROR_LOG entry, and any associated quarantine or slot incident record. Never include passwords, access tokens, or secret contents in support tickets.

End of Document