

1 ClickHouse CDC Production Readiness Guide

1.1 Debezium, Estuary Flow, and MaterializedPostgreSQL

Version boundary: Current documentation reviewed in August 2026. Product behavior, connector properties, ClickHouse Cloud support, and pricing can change. Verify the exact versions and deployment model before production rollout.

1.2 The decision in one page

PostgreSQL-to-ClickHouse CDC is not a single connector decision. It is a chain of source capture, durable state, transport, sink ingestion, target-table modeling, merge behavior, and query validation.

Choose this first when...	Candidate to evaluate	The decision is really about...
Kafka or a comparable event platform is already strategic and several consumers need the same stream	Debezium	Owning a reusable event backbone and its operational surface
The team wants managed capture and materialization with less self-managed streaming infrastructure	Estuary Flow	Moving infrastructure ownership to a managed service while retaining validation responsibility
ClickHouse is self-managed, the native path is currently supported, and a direct topology is valuable	MaterializedPostgreSQL	Accepting tighter coupling to PostgreSQL and ClickHouse support boundaries
ClickHouse Cloud is the destination	ClickPipes or another currently supported Cloud path	Verifying Cloud networking, source prerequisites, pricing, and edition-specific support

There is no universal winner. Score candidates using the workload you actually operate: number of tables, source write rate, UPDATE/DELETE ratio, initial snapshot size, schema-change frequency, number of downstream consumers, recovery objective, and query-visible freshness target.

Production rule: A pipeline is not ready because the connector is healthy. It is ready when the intended target state is query-visible, measurable, replay-safe, and recoverable after restart, outage, schema change, and source failover.

1.3 Decision scorecard

Use the workbook included in this kit. Give each criterion a weight from 1 to 5 and rate each candidate from 1 to 5. Multiply weight by rating; do not let the spreadsheet hide a veto condition. A candidate that fails a hard compatibility requirement should be marked **BLOCKED** even if its weighted score is high.

The most important criteria are deployment support, target correctness, initial-load behavior, schema evolution, recovery, observability, operational ownership, and total cost of ownership. Raw capture latency is useful, but it should not outweigh an inability to prove final query-visible state.

1.4 PostgreSQL preflight gate

Before starting a snapshot, record the PostgreSQL major version, hosting provider, database name, source tables, primary keys, replica identity, publication, replication-slot owner, expected write rate, longest transaction, and available storage for retained WAL.

The included `postgresql-preflight.sql` file separates read-only diagnostics from configuration templates. Run diagnostics first. Treat configuration statements as change-management examples, not commands to paste blindly into production.

Gate	Pass condition	Evidence to retain
Logical replication	The selected CDC path's source prerequisites are enabled and supported	Parameter output and provider-specific configuration
Publication	The exact tables and partitioned-table behavior are documented	Publication definition and table list
Identity	Each captured table has a stable key or an intentional replica-identity decision	Primary-key inventory and <code>REPLICA IDENTITY</code> output
Slot	Slot owner, restart LSN, confirmed flush LSN, active state, and invalidation status are known	<code>pg_replication_slots</code> snapshot
WAL budget	Normal and worst-case backlog fit the source storage budget	Write-rate, outage, and retained-WAL estimate
Permissions	The dedicated capture role has only the permissions required by the selected path	Role grants and provider policy
Failover	The recovery route and slot behavior after promotion are written and tested	Runbook, drill record, and endpoint plan

1.4.1 Replica identity is a deliberate choice

PostgreSQL normally uses a primary key as replica identity. `REPLICA IDENTITY FULL` can expose the old row image when a suitable key is not available or when the downstream design genuinely needs it. It can also increase logical WAL volume for `UPDATE` and `DELETE`-heavy workloads. Do not enable it across every table without measuring source impact and confirming that the sink needs the additional data.

```
-- PostgreSQL: use only after reviewing the table's key and CDC contract.
ALTER TABLE public.orders REPLICA IDENTITY FULL;

-- Restore the normal primary-key-based behavior when appropriate.
ALTER TABLE public.orders REPLICA IDENTITY DEFAULT;
```

1.5 Target-table contract in ClickHouse

ClickHouse is not an OLTP row store. A CDC target must define how inserts, updates, deletes, duplicate replays, and out-of-order events become an analytical state.

The following DDL is an illustrative starting point for a versioned current-state table. It is not a universal sink schema. The sink must actually map its event envelope into these columns, and the version must be comparable for all updates to the same key.

```
-- ClickHouse SQL: illustrative current-state CDC target.
CREATE TABLE default.orders_cdc
(
    id UInt64,
    user_id UInt64,
    status LowCardinality(String),
    total_amount Decimal(18, 2),
    updated_at DateTime64(3, 'UTC'),
    version UInt64,
    is_deleted UInt8
)
ENGINE = ReplacingMergeTree(version, is_deleted)
PRIMARY KEY id
ORDER BY id;
```

A versioned update is normally represented as a new row for the same sorting key. A delete can be represented as a newer tombstone row with `is_deleted = 1`, depending on the selected sink and target design.

```
-- Initial state.
INSERT INTO default.orders_cdc
    (id, user_id, status, total_amount, updated_at, version, is_deleted)
VALUES
    (42, 7, 'pending', 125.50, '2026-08-24 10:00:00.000', 1001, 0);

-- UPDATE becomes a newer version of the same key.
INSERT INTO default.orders_cdc
    (id, user_id, status, total_amount, updated_at, version, is_deleted)
VALUES
```

```

(42, 7, 'paid', 125.50, '2026-08-24 10:02:00.000', 1002, 0);

-- DELETE becomes a newer tombstone for the same key.
INSERT INTO default.orders_cdc
    (id, user_id, status, total_amount, updated_at, version, is_deleted)
VALUES
    (42, 7, 'paid', 125.50, '2026-08-24 10:03:00.000', 1003, 1);

-- FINAL forces query-time reconciliation for this read.
SELECT id, user_id, status, total_amount, version, is_deleted
FROM default.orders_cdc
FINAL
WHERE is_deleted = 0;

```

ReplacingMergeTree merges duplicate sorting-key rows asynchronously. Until a merge completes, a normal query can see more than one physical row for a key. FINAL can provide query-time reconciliation for a read, but it has a cost that must be measured. The correct design may instead use a materialized current-state table, a controlled query pattern, or a different sink/table model.

1.6 Debezium sink selection

Debezium captures source changes; the sink determines how records become ClickHouse rows. Decide what the event envelope looks like, which converter serializes it, how the topic maps to a table, and where update/delete semantics are enforced.

Sink path			Good starting condition	Verify before production
Official	ClickHouse	Kafka	Kafka Connect already exists and the team wants a ClickHouse-maintained sink path	Connector and server versions, converters, topic/table mapping, retries, DLQ behavior, and exactlyOnce scope
Altinity	ClickHouse	Sink Connector	The team wants an open-source alternative with documented Debezium CDC and ClickHouse-oriented mappings	Release compatibility, target-engine assumptions, data types, recovery, schema handling, and support boundary
ClickHouse Kafka table engine			ClickHouse should consume Kafka directly and the team will own parsing and materialized-view logic	Consumer groups, offsets, parsing failures, retries, raw envelope transformation, and target correctness

A sink's delivery guarantee is not the same as final table correctness. Test duplicate events, tombstones, late events, converter failures, schema changes, and rejected batches against the actual destination table.

1.7 Managed path: Estuary Flow

A managed service can reduce the amount of Kafka, Connect, and broker infrastructure your team assembles. It does not remove the need to understand source permissions, replication slots, backfills, destination behavior, schema policy, recovery, or cost.

Before committing, ask where captured state is retained, what happens during a destination outage, how a backfill restarts, what the service guarantees at capture and materialization boundaries, how fan-out changes cost, and how to export or recover data if the service is unavailable.

Use the included workbook to compare visible infrastructure savings with service fees, operator hours, source pressure, network transfer, retention, and recovery requirements.

1.8 Native path: MaterializedPostgreSQL

MaterializedPostgreSQL can reduce middleware when the current ClickHouse deployment supports the engine and the workload fits its boundaries. Current ClickHouse documentation describes a snapshot-plus-WAL logical-replication path, while also documenting limits around DDL, supported keys, replication slots, failover, TOAST, experimental status, and ClickHouse Cloud support.

The native path should therefore be evaluated as a deployment-specific compatibility decision, not as a default answer for all PostgreSQL-to-ClickHouse projects. Record the exact ClickHouse version, edition, PostgreSQL version, engine settings, schema shape, failover path, and reload procedure.

1.9 Data-type contract

Create a mapping table before the first snapshot. Do not wait for a conversion error to decide whether a timestamp is UTC, whether a JSON document is queryable or opaque, or whether a UUID should remain a native UUID.

PostgreSQL source	Conservative ClickHouse starting point	Main risk	Required test
timestampz	DateTime64(3, 'UTC') or a documented sink mapping	Session time zone and precision loss	Compare instants around DST, offsets, and millisecond boundaries
timestamp	DateTime64(3, 'UTC') only if the business contract defines it as UTC; otherwise preserve the stated zone contract	A zone-less value can be shifted incorrectly	Test source/session time zone and round-trip behavior
NUMERIC / DECIMAL	Decimal(P, S) sized for observed precision and scale	Overflow or rounding	Profile maximum precision, scale, nulls, and negative values

PostgreSQL source	Conservative ClickHouse starting point	Main risk	Required test
UUID	UUID when the sink and target support it; otherwise a documented string fallback	String fallback increases storage and changes validation	Compare values, null behavior, and serialization
JSONB	String or the current documented JSON type/path	Schema drift, nested extraction, and null semantics	Test representative documents, missing paths, arrays, and large payloads
BOOLEAN	Bool or the sink's documented integer mapping	Converter may emit boolean while target expects integer	Test true, false, null, and converter output
PostgreSQL arrays	Array(T) when the element mapping is supported	Nested/unsupported elements and null elements	Test empty arrays, null elements, and mixed historical values
BYTEA	A documented binary type or encoded String	Base64/hex expansion and semantic loss	Compare byte length and round-trip hashes

1.10 Initial-load and backfill gate

Initial load often dominates time to trust. Measure it separately from steady-state CDC. Record source CPU and I/O, connections, transaction duration, WAL growth, snapshot duration, first usable table, full catch-up, update/delete behavior during the snapshot, restart behavior, and final convergence.

A candidate fails the initial-load gate if it can stream quickly only after a snapshot that cannot complete within the source-storage or change-rate budget.

1.11 Four-checkpoint correctness test

Run the same test protocol for every candidate:

Checkpoint	Evidence	Pass condition
Source capture	LSN, source event, or managed capture status	The intended source change is exposed
Transport	Topic offset/record or managed durable state	The event remains recoverable
Sink ingestion	Insert count, sink metric, rejected-record log	ClickHouse accepted the intended event

Checkpoint	Evidence	Pass condition
Query-visible state	Key-level comparison after merge/reconciliation	The analytical query returns the intended state

The test dataset should contain inserts, multiple updates to one key, deletes, late events, duplicate replays, a schema addition, a rejected record, a destination outage, and a restart during the snapshot or catch-up window.

1.12 Observability and recovery

Monitor source slot backlog and WAL retention, connector or capture health, transport lag, sink errors, rejected records, ClickHouse insert failures, duplicate density, merge or freshness delay, schema-drift events, and last query-visible change per table.

A recovery plan should answer what happens when the destination is unavailable for one hour, when a slot is invalidated, when the connector restarts from an earlier position, when a source standby is promoted, when a schema change blocks a table, and when one table must be reloaded without rebuilding the entire pipeline.

1.13 Production go/no-go checklist

Mark each item **Pass**, **Fail**, or **Not applicable** and attach evidence.

Gate	Owner	Evidence	Status
Source logical replication and permissions reviewed			
Slot and WAL budget measured under outage assumptions			
Replica identity chosen per table			
Sink converter and event envelope documented			
ClickHouse target DDL reviewed and tested			
Initial snapshot/backfill completed within budget			
UPDATE and DELETE convergence proven			

Gate	Owner	Evidence	Status
Duplicate replay is harmless			
Schema-addition and incompatible-change behavior documented			
Destination outage and restart tested			
Source failover path tested or explicitly accepted			
Monitoring and alert owners assigned			
Rollback/reload procedure approved			
Cost model includes service, infrastructure, network, and operator time			

1.14 References