EcomLabs

CHECKLIST AND TESTED CODE

Adjust your metrics to the carts/update change

Since October 12, 2026, Shopify sends the carts/update webhook once per real cart change, not once per request. A checklist for counts, rates, alerts and year-over-year comparisons, plus an idempotent Node handler that stores each delivery once and separates deliveries from real changes.

FOR	Shopify apps that receive carts/update and count cart events
TESTED WITH	Node 22.22.3 · PGlite 0.5.10 (PostgreSQL 18.6) · Windows 11
UPDATED	November 11, 2026
DOCS	shopify.dev/changelog (October 12, 2026)
ARTICLE	ecomlabs.dev/g/carts-en
AUTHOR	Luis Morales · @luismoralesdev

Contents

Before you start	What changed
Step 01	Adjust metrics and reports
Step 02	The idempotent handler
Step 03	Deliveries vs. real changes
Step 04	On a development store
When you are done	Checklist

BEFORE YOU START

What changed

Per Shopify's changelog entry of October 12, 2026, a cart operation that leaves the cart unchanged no longer triggers `carts/update`: you used to get one webhook per request; now you get one per real change. It applies from that day at 10:00 a.m. Eastern (14:00 UTC), with no API version and no action required. Duplicates from retries still happen.

STEP 01

Adjust metrics and reports

Search your code and dashboards for every place that reads `carts/update`:

IF YOUR APP…	SINCE OCTOBER 12	WHAT TO DO
Counts `carts/update` deliveries	The count drops on the same traffic	Annotate the cutoff on the chart and in the report
Divides by `carts/update`	The rate goes up	Divide by real changes (Step 03)
Treats the webhook as an activity signal	An unchanged-cart request no longer arrives	Review abandoned cart timers
Has alerts with fixed thresholds	They can fire falsely	Recalibrate on post-cutoff data
Compares against 2025	Mixes two counting rules	Compare real changes, or leave cart activity out
Dedupes by `X-Shopify-Webhook-Id`	Retries still happen	Keep it

STEP 02

The idempotent handler

Save the handler as `cart-events.mjs`:

```
import { createHash, createHmac, timingSafeEqual } from "node:crypto";

export function verifyShopifyWebhook(rawBody, hmacHeader, secret) {
  if (!secret || typeof hmacHeader !== "string") return false;
  const expected = Buffer.from(createHmac("sha256", secret).update(rawBody).digest("base64"));
  const received = Buffer.from(hmacHeader);
  return expected.length === received.length && timingSafeEqual(expected, received);
}

// headers: lowercase names, as Node and the Fetch API give them.
export async function handleCartsUpdate(db, headers, rawBody, secret) {
```

```
  if (!verifyShopifyWebhook(rawBody, headers["x-shopify-hmac-sha256"], secret)) return { status: 401 };
  const webhookId = headers["x-shopify-webhook-id"];
  if (!webhookId) return { status: 400 };

  const { updated_at, ...cart } = JSON.parse(rawBody);
  const fingerprint = createHash("sha256").update(JSON.stringify(cart)).digest("hex");

  const { rows } = await db.query(
    `INSERT INTO cart_events (webhook_id, event_id, shop, cart_token, fingerprint, triggered_at)
     VALUES ($1, $2, $3, $4, $5, $6)
     ON CONFLICT (webhook_id) DO NOTHING
     RETURNING webhook_id`,
    [
      webhookId,
      headers["x-shopify-event-id"] ?? null,
      headers["x-shopify-shop-domain"],
      cart.token,
      fingerprint,
      headers["x-shopify-triggered-at"] ?? updated_at,
    ],
  );
  return { status: 200, counted: rows.length === 1 };
}
```

- **Signature.** An HMAC-SHA256 of the raw body (read it before any body parser) with your app's client secret. Invalid: `401`.
- **Duplicates.** `webhook_id` is the primary key and `ON CONFLICT DO NOTHING` skips a repeated id. A duplicate still gets `200`, as Shopify asks.
- **Order and fingerprint.** `triggered_at` comes from `X-Shopify-Triggered-At`, because Shopify does not guarantee order. The fingerprint is a hash of the cart without `updated_at`.

Create the table with `schema.sql`:

```
CREATE TABLE cart_events (
  webhook_id   text        PRIMARY KEY,
  event_id     text,
  shop         text        NOT NULL,
  cart_token   text        NOT NULL,
  fingerprint  text        NOT NULL,
  triggered_at timestamptz NOT NULL,
  received_at  timestamptz NOT NULL DEFAULT now()
);

CREATE INDEX cart_events_by_cart ON cart_events (shop, cart_token, triggered_at);
```

Test it: in the same folder, run `npm install @electric-sql/pglite` and save `dedupe.test.mjs`:

```
import { test } from "node:test";
import assert from "node:assert/strict";
```

```
import { createHmac } from "node:crypto";
import { readFileSync } from "node:fs";
import { PGlite } from "@electric-sql/pglite";
import { handleCartsUpdate } from "./cart-events.mjs";

const SECRET = "test_secret_do_not_use"; // placeholder, never your real secret
const body = JSON.stringify({ token: "exampleCartId", line_items: [{ key: "k1", quantity: 3 }], note: null });
const headers = {
  "x-shopify-hmac-sha256": createHmac("sha256", SECRET).update(body).digest("base64"),
  "x-shopify-shop-domain": "example.myshopify.com",
  "x-shopify-webhook-id": "wh-1",
  "x-shopify-triggered-at": "2026-10-11T18:00:00.000Z",
};

test("the same delivery five times counts once", async () => {
  const db = new PGlite();
  await db.exec(readFileSync(new URL("./schema.sql", import.meta.url), "utf8"));
  const results = [];
  for (let i = 0; i < 5; i++) results.push(await handleCartsUpdate(db, headers, body, SECRET));
  assert.deepEqual(results.map((r) => r.counted), [true, false, false, false, false]);
  assert.ok(results.every((r) => r.status === 200));
  const { rows } = await db.query("SELECT count(*)::int AS n FROM cart_events");
  assert.equal(rows[0].n, 1);
});
```

`node --test dedupe.test.mjs` printed `ok 1 - the same delivery five times counts once` and `pass 1` on Node 22.22.3 with PGlite 0.5.10: five identical deliveries, one row. As a control, a table without the primary key stored all five. The deliveries are simulated; we made no calls to Shopify.

STEP 03

Deliveries vs. real changes

Save `report.sql`. A delivery is a real change when its fingerprint differs from the previous one for the same cart:

```
SELECT day,
       count(*)                                          AS deliveries,
       count(*) FILTER (WHERE prev IS DISTINCT FROM fp) AS real_changes
FROM (
  SELECT (triggered_at AT TIME ZONE 'America/Mexico_City')::date AS day,
         fingerprint AS fp,
         lag(fingerprint) OVER (PARTITION BY shop, cart_token ORDER BY triggered_at) AS prev
  FROM cart_events
) e
GROUP BY day
ORDER BY day;
```

`deliveries` is what your app used to see; `real_changes` is the series you can compare across the cutoff. Swap the time zone for the one your reports use. With test deliveries from October 11 (one retry, one unchanged cart) and

October 13, arriving out of order, it returned 3 deliveries and 2 real changes on the 11th, and 1 and 1 on the 13th.

> **CAUTION**
>
> This is an approximation: the sample payload does not include cart attributes, so a change to attributes alone looks like no change. If you change `api_version` or `include_fields`, every cart will show one extra change that day. And PGlite runs one query at a time: repeat the simultaneous-duplicates test on a real PostgreSQL server, never on production.

STEP 04

On a development store

We did not run this part, because it needs a real store and account. `shopify app webhook trigger` cannot reproduce the change, since it always sends the same sample payload. Subscribe the app in `shopify.app.toml` and run `shopify app dev`:

```
[access_scopes]
scopes = "read_orders"

[webhooks]
api_version = "2026-10"

[[webhooks.subscriptions]]
topics = ["carts/update"]
uri = "/webhooks/carts"
```

Add a product in the online store. From the browser console, send the line's `key` to `/cart/change.js` with the same quantity: no `carts/update` should arrive. Raise it by one: exactly one should arrive. The console code is in the article. Watch the deliveries in the Dev Dashboard (Apps, your app, Logs), which can lag by several minutes.

WHEN YOU ARE DONE

Checklist

- [ ] Every dashboard marks the October 12, 2026, 10:00 a.m. ET cutoff and does not use that mixed day as a baseline.
- [ ] The handler verifies the signature and stores each delivery once, with `X-Shopify-Webhook-Id` as the primary key.
- [ ] Reports sort by `triggered_at` and separate deliveries from real changes.
- [ ] Only one `carts/update` subscription writes to the table (or you count by `event_id`).
- [ ] Comparisons against 2025 use real changes or leave cart activity out.
- [ ] Alerts are recalibrated, and your merchants know why their dashboards shifted.