EcomLabs

CHECKLIST Y CÓDIGO PROBADO

Ajusta tus métricas al cambio de carts/update

Desde el 12 de octubre de 2026, Shopify envía el webhook carts/update una vez por cambio real del carrito, no por petición. La checklist para revisar conteos, tasas, alertas y comparativos, y un handler idempotente en Node que guarda cada entrega una sola vez y separa entregas de cambios reales.

PARA	Apps de Shopify que reciben carts/update y cuentan eventos de carrito
PROBADO CON	Node 22.22.3 · PGlite 0.5.10 (PostgreSQL 18.6) · Windows 11
ACTUALIZADA	11 de noviembre de 2026
DOCUMENTACIÓN	shopify.dev/changelog (12 de octubre de 2026)
ARTÍCULO	ecomlabs.dev/g/carts
AUTOR	Luis Morales · @luismoralesdev

Índice

Antes de empezar	Qué cambió
Paso 01	Ajusta métricas y reportes
Paso 02	El handler idempotente
Paso 03	Entregas contra cambios reales
Paso 04	En una tienda de desarrollo
Al terminar	Checklist

ANTES DE EMPEZAR

Qué cambió

Según el changelog de Shopify del 12 de octubre de 2026, una operación que deja el carrito igual ya no dispara `carts/update`: antes llegaba un webhook por petición; ahora, uno por cambio real. Rige desde ese día a las 10:00 hora del Este (08:00 en la Ciudad de México), sin versión de la API ni acción requerida. Los duplicados por reintento siguen.

PASO 01

Ajusta métricas y reportes

Busca en tu código y en tus tableros cada lugar que lee `carts/update`:

SI TU APP…	DESDE EL 12 DE OCTUBRE	QUÉ HACER
Cuenta entregas de `carts/update`	El conteo baja con el mismo tráfico	Anota el corte en la gráfica y en el reporte
Divide entre `carts/update`	La tasa sube	Divide entre cambios reales (Paso 03)
Usa el webhook como señal de actividad	Una petición sin cambios ya no llega	Revisa temporizadores de carrito abandonado
Tiene alertas con umbrales fijos	Pueden dispararse en falso	Recalibra con datos posteriores al corte
Compara contra 2025	Mezcla dos reglas de conteo	Compara cambios reales o deja fuera la actividad de carrito
Deduplica por `X-Shopify-Webhook-Id`	Los reintentos siguen	Mantenlo

PASO 02

El handler idempotente

Guarda el handler en `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 };
}
```

- **Firma.** HMAC-SHA256 del cuerpo crudo (léelo antes de cualquier parser) con el client secret de tu app. Inválida: `401`.
- **Duplicados.** `webhook_id` es la clave primaria y `ON CONFLICT DO NOTHING` no inserta un id repetido. Un duplicado recibe `200`, como pide Shopify.
- **Orden y huella.** `triggered_at` viene de `X-Shopify-Triggered-At`, porque Shopify no garantiza el orden. La huella es un hash del carrito sin `updated_at`.

Crea la tabla con `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);
```

Pruébalo: en la misma carpeta, corre `npm install @electric-sql/pglite` y guarda `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` imprimió `ok 1 - the same delivery five times counts once` y `pass 1` con Node 22.22.3 y PGlite 0.5.10: cinco entregas iguales, una fila. Como control, una tabla sin clave primaria guardó las cinco. Las entregas son simuladas; no llamamos a Shopify.

PASO 03

Entregas contra cambios reales

Guarda `report.sql`. Una entrega es un cambio real si su huella es distinta de la anterior del mismo carrito:

```
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` es lo que tu app veía; `real_changes` es la serie que puedes comparar a través del corte. Cambia la zona horaria por la de tus reportes. Con entregas de prueba del 11 de octubre (un reintento y una sin cambios) y del

13, llegadas en desorden, dio 3 entregas y 2 cambios reales el 11, y 1 y 1 el 13.

> **CUIDADO**
>
> Es una aproximación: el payload de ejemplo no trae los atributos del carrito, así que un cambio solo en atributos se ve como «sin cambio». Si cambias `api_version` o `include_fields`, cada carrito mostrará un cambio extra ese día. Y PGlite ejecuta una consulta a la vez: repite la prueba de duplicados simultáneos en un PostgreSQL de verdad, nunca en producción.

PASO 04

En una tienda de desarrollo

Esta parte no la corrimos: requiere una tienda y una cuenta reales. `shopify app webhook trigger` no reproduce el cambio, porque siempre manda el mismo payload de ejemplo. Suscribe la app en `shopify.app.toml` y corre `shopify app dev`:

```
[access_scopes]
scopes = "read_orders"

[webhooks]
api_version = "2026-10"

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

Agrega un producto en la tienda online. Desde la consola del navegador, manda a `/cart/change.js` la `key` de la línea con la misma cantidad: no debería llegar ningún `carts/update`. Súbela en uno: debería llegar exactamente uno. El código para la consola está en el artículo. Mira las entregas en el Dev Dashboard (Apps, tu app, Logs), que puede tardar varios minutos.

AL TERMINAR

Checklist

- [ ] Cada tablero marca el corte del 12 de octubre de 2026, 10:00 ET, y no usa ese día mixto como base.
- [ ] El handler verifica la firma y guarda cada entrega una vez, con `X-Shopify-Webhook-Id` como clave primaria.
- [ ] Los reportes ordenan por `triggered_at` y separan entregas de cambios reales.
- [ ] Una sola suscripción a `carts/update` escribe en la tabla (o cuentas por `event_id`).
- [ ] Los comparativos contra 2025 usan cambios reales o dejan fuera la actividad de carrito.
- [ ] Recalibraste las alertas y avisaste a tus clientes por qué cambiaron sus tableros.