EcomLabs

GUÍA DE MIGRACIÓN

Migrar un descuento de Shopify Scripts a Functions

Una promoción por volumen que vivía en un Script de Ruby, reescrita como Function de descuento en JavaScript: input query, función, shopify.extension.toml, cinco carritos de prueba con su salida real, la mutación para crear el descuento y cómo apilarlo con otro en la misma línea.

PARA	Tiendas Shopify Plus que tenían promociones en Shopify Scripts y apps que las migran
PROBADO CON	Node 22.22.3 · graphql 16.15.0 (esquemas de Discount Function 2026-04 y 2026-07) · Windows 11
ACTUALIZADA	11 de noviembre de 2026
DOCUMENTACIÓN	shopify.dev/docs/api/functions/latest/discount
ARTÍCULO	ecomlabs.dev/g/functions
AUTOR	Luis Morales · @luismoralesdev

Índice

Antes de empezar	Qué vas a migrar
Paso 01	Genera la extensión
Paso 02	La input query
Paso 03	La función
Paso 04	Prueba en Node con cinco carritos
Paso 05	Crea el descuento en una tienda de desarrollo
Al terminar	Checklist

ANTES DE EMPEZAR

Qué vas a migrar

Desde el 30 de junio de 2026 no se ejecuta ningún Shopify Script (changelog de Shopify del 9 de abril de 2026). Esta guía rehace una promoción como Function de descuento en JavaScript, con la plantilla de la Shopify CLI y la Discount Function API 2026-10: productos con la etiqueta `bulk-tier` tienen 10% con 3 a 5 unidades en la línea y 15% con 6 o más, nunca tarjetas de regalo, y se combina con un descuento de lealtad en la misma línea. El Script original, que reescribía el precio con `change_line_price`, está en el artículo.

PASO 01

Genera la extensión

Desde la carpeta de tu app:

```
shopify app generate extension --template discount --flavor vanilla-js --name volume-discount
```

La plantilla trae dos targets. Esta promoción solo usa el de líneas: borra el target `cart.delivery-options.discounts.generate.run` del TOML, sus archivos `src/cart_delivery_options*` y su línea en `src/index.js`.

En el TOML que generó la CLI, pon `api_version = "2026-10"`, deja `handle = "volume-discount"` y conserva un solo bloque `[[extensions.targeting]]`: `target = "cart.lines.discounts.generate.run"`, `input_query = "src/cart_lines_discounts_generate_run.graphql"` y `export = "cart-lines-discounts-generate-run"`. `[extensions.build]` queda como viene (`path = "dist/function.wasm"`). El `handle` es el `functionHandle` del paso 05.

PASO 02

La input query

Reemplaza `src/cart_lines_discounts_generate_run.graphql`:

```
query CartInput {
  cart {
    lines {
      id
      quantity
      merchandise { __typename ... on ProductVariant { product { isGiftCard hasAnyTag(tags: ["bulk-tier"]) } } }
}
    }
  }
  discount { discountClasses metafield(namespace: "$app", key: "volume-tiers") { jsonValue } }
}
```

`hasAnyTag` devuelve un booleano: las etiquetas no se leen como lista. El metafield del descuento guarda los niveles.

PASO 03

La función

Reemplaza `src/cart_lines_discounts_generate_run.js`:

```
// The tiers the Script had as constants. A "volume-tiers" metafield on the discount overrides them.
const DEFAULT_TIERS = [
  { minQuantity: 6, percentage: 15 },
  { minQuantity: 3, percentage: 10 },
];

export function cartLinesDiscountsGenerateRun(input) {
  if (!input.discount.discountClasses.includes("PRODUCT")) {
    return { operations: [] };
  }

  const tiers = [...(input.discount.metafield?.jsonValue?.tiers ?? DEFAULT_TIERS)]
    .sort((a, b) => b.minQuantity - a.minQuantity);

  const candidates = [];
  for (const line of input.cart.lines) {
    if (line.merchandise.__typename !== "ProductVariant") continue;
    const { product } = line.merchandise;
    if (product.isGiftCard || !product.hasAnyTag) continue;

    const tier = tiers.find((t) => line.quantity >= t.minQuantity);
    if (!tier) continue;

    candidates.push({
      message: `${tier.percentage}% off for ${tier.minQuantity}+`,
      targets: [{ cartLine: { id: line.id } }],
      value: { percentage: { value: tier.percentage.toFixed(1) } },
    });
  }

  if (candidates.length === 0) {
    return { operations: [] };
  }

  return {
    operations: [{ productDiscountsAdd: { candidates, selectionStrategy: "ALL" } }],
  };
}
```

El porcentaje va como texto (`"15.0"`) y `ALL` aplica todos los candidatos (con `FIRST`, solo el primero). La plantilla importa estos valores desde `../generated/api`; aquí van como texto para correr en Node.

PASO 04

Prueba en Node con cinco carritos

En la carpeta de la extensión, crea `inputs/cart-mixed.json`, con la forma exacta que devuelve la query:

```
{ "cart": { "lines": [
  { "id": "gid://shopify/CartLine/1", "quantity": 6, "merchandise": { "__typename": "ProductVariant",
    "product": { "isGiftCard": false, "hasAnyTag": true } } },
  { "id": "gid://shopify/CartLine/2", "quantity": 3, "merchandise": { "__typename": "ProductVariant",
    "product": { "isGiftCard": false, "hasAnyTag": true } } },
  { "id": "gid://shopify/CartLine/3", "quantity": 10, "merchandise": { "__typename": "ProductVariant",
    "product": { "isGiftCard": true, "hasAnyTag": true } } },
  { "id": "gid://shopify/CartLine/4", "quantity": 1, "merchandise": { "__typename": "ProductVariant",
    "product": { "isGiftCard": false, "hasAnyTag": true } } },
  { "id": "gid://shopify/CartLine/5", "quantity": 8, "merchandise": { "__typename": "CustomProduct" } }
] },
  "discount": { "discountClasses": ["PRODUCT"], "metafield": null } }
```

Los otros cuatro son copias de ese archivo con cambios:

- `cart-qualifies.json`: solo la línea 1, con `"quantity": 4`.
- `cart-no-match.json`: la línea 1 con `"quantity": 2` y la línea 2 con `"quantity": 5` y `"hasAnyTag": false`.
- `cart-config.json`: solo la línea 1, con `"quantity": 2` y `"metafield": { "jsonValue": { "tiers": [{ "minQuantity": 2, "percentage": 5 }] } }`.
- `cart-order-only.json`: solo la línea 1 y `"discountClasses": ["ORDER"]`.

Guarda `run-carts.mjs` junto a `package.json` (la plantilla ya trae `"type": "module"`) y córrelo:

```
import { readdirSync, readFileSync } from "node:fs";
import { cartLinesDiscountsGenerateRun } from "./src/cart_lines_discounts_generate_run.js";

for (const file of readdirSync("inputs")) {
  const input = JSON.parse(readFileSync(`inputs/${file}`, "utf8"));
  const { operations } = cartLinesDiscountsGenerateRun(input);
  const candidates = operations[0]?.productDiscountsAdd.candidates ?? [];
  console.log(file, candidates.map((c) => `${c.targets[0].cartLine.id} ${c.value.percentage.value}%`));
}

$ node run-carts.mjs
cart-config.json [ 'gid://shopify/CartLine/1 5.0%' ]
cart-mixed.json [ 'gid://shopify/CartLine/1 15.0%', 'gid://shopify/CartLine/2 10.0%' ]
cart-no-match.json []
cart-order-only.json []
cart-qualifies.json [ 'gid://shopify/CartLine/1 10.0%' ]
```

Esa es la salida real con Node 22.22.3. También validamos la query y cada salida contra los esquemas 2026-04 y 2026-07. Dentro de tu app, el siguiente paso es la versión compilada: `shopify app function run --input inputs/cart-mixed.json --export cart-lines-discounts-generate-run` (no lo ejecutamos: no creamos ninguna app).

PASO 05

Crea el descuento en una tienda de desarrollo

Crea una tienda de desarrollo con el plan Plus desde el Dev Dashboard, corre `shopify app dev`, presiona `g` para abrir GraphiQL, elige la versión estable más reciente y ejecuta:

```
mutation CreateVolumeDiscount {
  discountAutomaticAppCreate(automaticAppDiscount: {
    title: "Volume discount"
    functionHandle: "volume-discount"
    discountClasses: [PRODUCT]
    startsAt: "2026-11-11T00:00:00Z"
    tags: ["volume"]
    combinesWith: { productDiscounts: true, productDiscountsWithTagsOnSameCartLine: { add: ["loyalty"] } }
    metafields: [{ namespace: "$app", key: "volume-tiers", type: "json",
      value: "{\"tiers\":[{\"minQuantity\":6,\"percentage\":15},{\"minQuantity\":3,\"percentage\":10}]}" }]
  }) {
    automaticAppDiscount { discountId status combinesWith { productDiscounts
productDiscountsWithTagsOnSameCartLine } }
    userErrors { field message }
  }
}
```

`tags` son las etiquetas propias del descuento; `add`, las que acepta en la misma línea. Actualizar `tags` reemplaza las anteriores (para sumar, `tagsAdd`). Si `userErrors` dice «Could not find Function», revisa que `app dev` siga corriendo, el handle y el permiso `write_discounts`.

> **CUIDADO**
>
> Apilar en la misma línea es solo Shopify Plus (Admin API 2026-04). Validamos esta mutación contra el esquema, pero no la ejecutamos en ninguna tienda.

Apila el descuento de lealtad y prueba

1. En el admin, crea un descuento de producto (por ejemplo, $20 en camisetas) con la etiqueta `loyalty`.
2. En **Combinations**: **Product discounts**, **Multiple per product** y, en **Combine on same product with discounts tagged**, `volume`.
3. Desactiva los demás descuentos, prueba 2 unidades (nada), 4 (10%), 6 (15%) y 6 con el de lealtad (los dos; el porcentaje va antes que el monto fijo), y repite cada ejecución con `shopify app function replay`.

AL TERMINAR

Checklist

- [ ] Cada promoción que vivía en Scripts está escrita en prosa, con sus combinaciones.
- [ ] Los cinco carritos dan la salida esperada en Node.
- [ ] Los dos descuentos tienen etiquetas recíprocas y la tienda es Plus.