# Build Your Own Marketing Dashboard

Connect Shopify, Meta Ads & Google Analytics
into one real-time dashboard — built with Claude

Shopify | Meta Ads | Google Analytics | Vercel

# Build Your Own Marketing Dashboard

This guide walks you through building a custom marketing dashboard that pulls live data from Shopify, Meta Ads, and Google Analytics 4 — all in one place, with automatic year-over-year comparisons, custom date ranges, and a clean UI your whole team can use.

No expensive third-party tools. No monthly SaaS fees. Just a lightweight app you build once with Claude, deploy free on Vercel, and own forever. The whole build takes a few hours and Claude handles all the code.

> ★ *Why build your own? Tools like Triple Whale or Northbeam cost $300–$1,000+/month. This dashboard costs $0 to run and shows exactly the metrics you actually care about.*

**What you'll build**

- ✓ A live dashboard showing revenue, ROAS, sessions, CVR, and new vs returning customers
- ✓ Shopify connected via OAuth — full order history with pagination
- ✓ Meta Ads connected via permanent system user token — spend, ROAS, cost per result, TACOS
- ✓ Google Analytics 4 connected via OAuth refresh token — sessions, CVR, channel breakdown
- ✓ Custom date range picker with automatic year-over-year comparison
- ✓ Quick presets: This Month, Last Month, Q4, YTD, Last Year
- ✓ Deployed free on Vercel with 24-hour edge caching for fast loads
- ✓ Mobile responsive — works on any device

| Section | Data source | Key metrics |
|---|---|---|
| Revenue | Shopify | GMV, orders, AOV, new vs returning customers |
| Traffic & Conversion | Google Analytics 4 | Sessions, CVR, new users, channel breakdown |
| META Ads | Meta Marketing API | Ad spend, ROAS, cost per result, TACOS |
| Year-over-year | All sources | Auto-calculated — same date range, -365 days |

# Before You Start

You need access to four things before opening Claude. None of these cost money to set up.

| Requirement | What it is | Where to get it |
|---|---|---|
| Shopify store | Your ecommerce store with admin access | Your existing Shopify account |
| Meta Business | Business Manager with your ad account | business.facebook.com |
| Google Analytics 4 | GA4 property connected to your site | analytics.google.com |
| Vercel account | Free hosting for your dashboard | vercel.com — free plan works |
| Node.js | JavaScript runtime on your Mac/PC | nodejs.org — download LTS |
| GitHub account | Required for Vercel deployment | github.com — free account |

■ *Claude does all the coding. You don't need to write a single line of JavaScript. Your job is to follow the steps, paste credentials, and test that data loads correctly.*

# 1 Create the Project

Scaffold the app and folder structure

Start by asking Claude to scaffold the entire project. This creates your folder structure, package.json, and Vercel config in one shot.

**■ PROMPT — Copy and paste into Claude**

```
I want to build a marketing dashboard web app that connects to Shopify,

Meta Ads, and Google Analytics 4.



Please scaffold a complete project with:

- A /api/ folder with shopify.js, ga.js, and meta.js proxy files

- A /public/index.html frontend dashboard

- A package.json with necessary dependencies

- A vercel.json config with 60s function timeout and API rewrites

- A .env.local template listing all environment variables I'll need



The dashboard should have sections for Revenue, Traffic & Conversion,

and META Ads, with a custom date range picker and year-over-year comparison.

Make it mobile responsive with a clean minimal design.
```

## Project structure Claude will create

```
my-dashboard/

■■■ api/

■ ■■■ shopify.js ← Shopify Admin API proxy

■ ■■■ ga.js ← GA4 proxy

■ ■■■ meta.js ← Meta Marketing API proxy

■■■ public/

■ ■■■ index.html ← Dashboard frontend

■■■ package.json

■■■ vercel.json ← 60s timeout + rewrites

■■■ .env.local ← Your credentials (never committed to git)
```

## Install and test locally

Run each command separately in Terminal:

**Step 1 — Install dependencies**

```
cd ~/Desktop/my-dashboard && npm install
```

**Step 2 — Start local server**

```
npx vercel dev
```

Then open **http://localhost:3000** in your browser.

## Step 2 — Shopify

Connect your store via OAuth app

Shopify killed legacy custom app tokens on January 1, 2026. You now need to create a proper OAuth app through the Shopify Developer Dashboard. Claude will build the full connection including auto token refresh.

### 2.1 Create your Shopify OAuth app

- Go to **partners.shopify.com** → Apps → Create app
- Choose **Public app** → name it *My Dashboard*
- Under API credentials, copy your **Client ID** and **Client Secret**
- Set required API scopes: **read_orders, read_products, read_customers**
- Install the app on your store and copy the resulting **access token**

### 2.2 Ask Claude to build the Shopify API proxy

**■ PROMPT — Copy and paste into Claude**

```
Build me a Shopify API proxy at api/shopify.js that:

1. Accepts ?from=YYYY-MM-DD&to;=YYYY-MM-DD query parameters
2. Also accepts ?yoyFrom= and ?yoyTo= for year-over-year comparison
3. Fetches all orders using full pagination (follow the Link header)
4. Calculates: total revenue, order count, AOV, new vs returning customers
- Revenue = total_line_items_price - total_discounts
- New customer = customer.created_at is within the period
5. Auto-refreshes the access token using client_credentials grant
(falls back to static SHOPIFY_TOKEN if refresh fails)
6. Returns current period and YoY data in a single JSON response
7. Adds 24hr Vercel edge cache headers

Environment variables: SHOPIFY_DOMAIN, SHOPIFY_TOKEN,
SHOPIFY_CLIENT_ID, SHOPIFY_CLIENT_SECRET
```

## 2.3 Environment variables

| Variable | Description |
|---|---|
| `SHOPIFY_DOMAIN` | yourstore.myshopify.com |
| `SHOPIFY_TOKEN` | Static fallback access token (shpat_…) |
| `SHOPIFY_CLIENT_ID` | OAuth client ID for auto token refresh |
| `SHOPIFY_CLIENT_SECRET` | OAuth client secret for auto token refresh |

## 2.4 Test it

Paste this into Terminal to confirm Shopify is returning data:

**Test Shopify API**

```
curl "http://localhost:3000/api/shopify?from=2026-01-01&to=2026-01-31"
```

■ *You should see a JSON response with revenue, orderCount, aov, newCustomers, returningCustomers. If you get a 401 error, double-check your SHOPIFY_DOMAIN includes .myshopify.com and your token has read_orders scope.*

## Step 3 — Meta Ads

Connect via permanent system user token

Meta's regular user tokens expire every 60 days — which breaks your dashboard. The fix is a **system user token** that never expires. This takes about 15 minutes to set up in Meta Business Manager.

### 3.1 Create a system user in Meta Business Manager

- Go to **business.facebook.com** → Settings → Users → System Users
- Click **Add** → name it *Dashboard* → Role: **Admin**
- Click **Generate New Token** → select your app
- Scopes needed: **ads_read, ads_management, business_management**
- Set token expiry to **Never** → Generate → copy the token immediately
- Assign the system user to your Ad Account with **Analyst** access

> ■ *Important: Copy this token immediately and save it in your .env.local file. Meta only shows it once and you cannot retrieve it again.*

### 3.2 Find your Ad Account ID

- In Meta Business Manager → Ad Accounts
- Your Ad Account ID looks like: **act_1234567890**
- Copy the full string including the **act_** prefix

### 3.3 Ask Claude to build the Meta API proxy

■ **PROMPT — Copy and paste into Claude**

```
Build me a Meta Ads API proxy at api/meta.js that:



1. Accepts ?from=YYYY-MM-DD&to;=YYYY-MM-DD and ?yoyFrom= ?yoyTo= parameters

2. Queries the Meta Marketing API v18.0 for the ad account

3. Returns for each period: ad spend, ROAS, cost per result, TACOS

- ROAS = purchase_roas from action_values

- Cost per result = spend / purchases (from actions)

- TACOS = spend / Shopify revenue (pass as a query param)

4. Fetches current and YoY periods in parallel

5. Adds 24hr Vercel edge cache headers



Environment variables: META_ACCESS_TOKEN, META_AD_ACCOUNT_ID
```

## 3.4 Environment variables

| Variable | Description |
|---|---|
| `META_ACCESS_TOKEN` | Permanent system user token (never expires) |
| `META_AD_ACCOUNT_ID` | Your ad account ID including act_ prefix |

## 3.5 Test it

**Test Meta API**

```
curl "http://localhost:3000/api/meta?from=2026-01-01&to=2026-01-31"
```

## Step 4 — Google Analytics 4

Connect via OAuth refresh token

GA4 requires OAuth authentication. You'll set up a Google Cloud project, create OAuth credentials, and generate a refresh token that never expires. Claude handles all the API calls once you have the token.

### 4.1 Set up Google Cloud credentials

- Go to **console.cloud.google.com** → New Project → name it *My Dashboard*
- Enable the **Google Analytics Data API**
- Go to **APIs & Services** → **OAuth consent screen** → External → fill in app name
- Go to **Credentials** → **Create Credentials** → **OAuth 2.0 Client ID**
- Application type: **Web application**
- Authorized redirect URI: **https://developers.google.com/oauthplayground**
- Copy your **Client ID** and **Client Secret**

### 4.2 Generate your refresh token (one time only)

- Go to **developers.google.com/oauthplayground**
- Click the settings gear → check **Use your own OAuth credentials**
- Enter your Client ID and Client Secret
- In Step 1, find and select: **Google Analytics Data API v1beta**
- Click **Authorize APIs** → sign in with your Google account
- Click **Exchange authorization code for tokens**
- Copy the **Refresh token** — this never expires

### 4.3 Ask Claude to build the GA4 proxy

■ **PROMPT — Copy and paste into Claude**

```
Build me a Google Analytics 4 API proxy at api/ga.js that:



1. Accepts ?from=YYYY-MM-DD&to;=YYYY-MM-DD and ?yoyFrom= ?yoyTo= parameters

2. Uses OAuth refresh token to get a fresh access token on each request

3. Makes 3 parallel GA4 API calls for each period (6 calls total):

- Report 1: sessions, conversions, conversion rate, new users

- Report 2: sessions by channel (Organic, Paid, Direct, Email, etc.)

- Report 3: top 10 pages by sessions

4. Returns all data in a single JSON response

5. Adds 24hr Vercel edge cache headers



Environment variables:

GA_PROPERTY_ID, GA_CLIENT_ID, GA_CLIENT_SECRET, GA_REFRESH_TOKEN
```

## 4.4 Environment variables

| Variable | Description |
|---|---|
| `GA_PROPERTY_ID` | Your GA4 property ID (numbers only, no 'properties/') |
| `GA_CLIENT_ID` | Google OAuth client ID |
| `GA_CLIENT_SECRET` | Google OAuth client secret |
| `GA_REFRESH_TOKEN` | OAuth refresh token from playground — never expires |

## 4.5 Test it

**Test GA4 API**

```
curl "http://localhost:3000/api/ga?from=2026-01-01&to=2026-01-31"
```

# 5 Build the Dashboard UI

Date picker, metric cards, YoY comparison

With all three API proxies working, ask Claude to build the full frontend. This is a single HTML file that calls your APIs and renders everything in a clean, responsive layout.

**■ PROMPT — Copy and paste into Claude**

```
Build me a complete dashboard frontend at public/index.html that:



1. Has a custom date range picker (From / To date inputs)

2. Has quick preset buttons: This Month, Last Month, Q4, YTD, Last Year

3. Auto-calculates YoY comparison (same range shifted back 365 days)

4. Has 4 sections: Revenue, Traffic & Conversion, Customers, META Ads

5. Each metric card shows: current value, YoY value, and % change

(green up arrow if improved, red down arrow if declined)

6. Revenue section: GMV, orders, AOV

7. Traffic section: sessions, CVR, new users, channel breakdown table

8. Customers section: new customers, returning customers, % new

9. META Ads section: ad spend, ROAS, cost per result, TACOS

10. Shows an error banner if any API fails (with sample data fallback)

11. Mobile responsive — stacks to single column on small screens

12. Clean minimal design — dark header, white cards, subtle shadows



It should call /api/shopify, /api/ga, and /api/meta with the selected dates.
```

## Key UI features to verify

| Feature | What to check |
|---|---|

| | |
|---|---|
| Date range picker | Changing dates triggers new API calls and updates all cards |
| YoY comparison | Each card shows both current and prior year numbers |
| % change indicators | Green ↑ for improvements, red ↓ for declines |
| Error banner | If an API fails, banner shows which source is on sample data |
| Mobile layout | Resize browser — cards should stack to single column |
| Loading state | Cards show a loading indicator while APIs fetch data |

## Step 6 — Deploy to Vercel

Free hosting, live in under 2 minutes

Vercel is free for personal projects and deploys in seconds. Your dashboard will get a public URL you can share with your team immediately.

### 6.1 Install Vercel CLI (one time only)

**Install Vercel CLI globally**

```
npm install -g vercel
```

### 6.2 Navigate to your project folder

**Go to your project**

```
cd ~/Desktop/my-dashboard
```

### 6.3 Deploy to production

**Deploy — Vercel will ask a few setup questions the first time**

```
npx vercel --prod
```

When prompted: No to existing project, set name to **my-dashboard**, press Enter for directory.

### 6.4 Add environment variables in Vercel

- Go to **vercel.com** → your project → **Settings** → **Environment Variables**
- Add every variable from your .env.local file one by one
- Set environment to **Production** for all of them

### 6.5 Redeploy after adding env vars

**Redeploy so Vercel picks up your environment variables**

```
npx vercel --prod
```

## 6.6 Test each API on your live URL

Replace **your-dashboard.vercel.app** with your actual Vercel URL:

**Test Shopify live**

```
curl "https://your-dashboard.vercel.app/api/shopify?from=2026-01-01&to=2026-01-31"
```

**Test GA4 live**

```
curl "https://your-dashboard.vercel.app/api/ga?from=2026-01-01&to=2026-01-31"
```

**Test Meta live**

```
curl "https://your-dashboard.vercel.app/api/meta?from=2026-01-01&to=2026-01-31"
```

■ *First load after deploy is slow (Vercel cold start — 3-5 seconds). Subsequent loads are fast due to 24hr edge caching. Share the URL with your team once all three APIs return real data.*

# Troubleshooting

| Problem | Fix |
|---|---|
| Shopify returns 401 | Token expired. Generate a new access token from Shopify Partners and update SHOPIFY_TOKEN in Vercel env vars. |
| Meta returns error 190 | You used a user token instead of a system user token — it expired after 60 days. Create a system user token (never expires). |
| GA4 returns 403 | GA4 Data API not enabled. Go to Google Cloud Console → APIs & Services → Library → enable Google Analytics Data API. |
| Dashboard shows sample data | One API is failing. Open browser DevTools → Network tab → look for red API calls to see the error message. |
| Vercel timeout on first load | Normal cold start. If it persists, confirm vercel.json has maxDuration: 60 set for API functions. |
| YoY numbers look wrong | Check date math — YoY must be exactly 365 days back, not 1 calendar year (matters for leap years). |
| If Shopify token auto-refresh breaks | Manually generate a new token with the curl command below, then update SHOPIFY_TOKEN in Vercel. |

## Manually refresh your Shopify token

If auto-refresh ever stops working, use this command to get a new token:

**Get a new Shopify token manually (replace yourstore, YOUR_ID, YOUR_SECRET)**

```
curl -X POST https://yourstore.myshopify.com/admin/oauth/access_token \ -H
"Content-Type: application/x-www-form-urlencoded" \ -d
"grant_type=client_credentials&client_id=YOUR_ID&client_secret=YOUR_SECRET"
```

---

## What to build next

Once your core dashboard is running, these are the highest-value additions:

| | |
|---|---|
| → **Amazon Seller Central** | Add SP-API integration for Amazon revenue and orders alongside Shopify. |
| → **Email alerts** | Ask Claude to add a daily email digest — revenue vs prior day, flagged anomalies. |
| → **Slack notifications** | Post daily or weekly summaries to your team's Slack channel automatically. |
| → **Custom domain** | Connect a domain like dashboard.yourbrand.com via Vercel's domain settings. |

---

*Guide created 2026 | Whiskey & Pack | whiskeyandpack.com | Built with Claude AI*