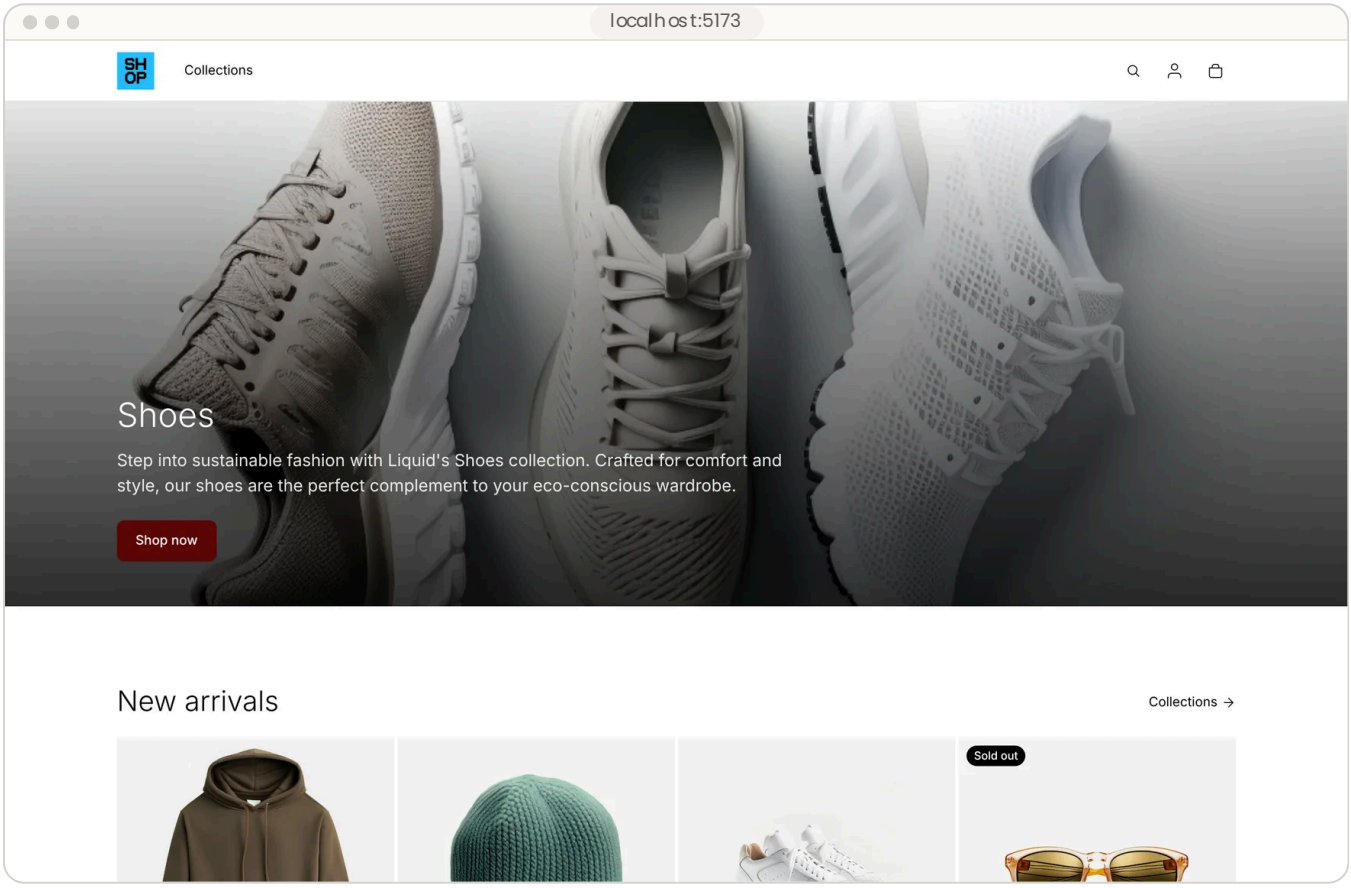
Lass Claude Code deinen Shopify-Shop umbauen und spar dir die App-Gebühren

Das Briefing für Claude und die Einrichtung in sieben Schritten: Hydrogen, GitHub, ein eigener Zugang und ein Vorschau-Link vor jedem Livegang.

 **Berkan Ceyhan** · Stand Oktober 2026



Die Hydrogen-Vorlage von Shopify, lokal gestartet. Die Produkte sind Demodaten von mock.shop.

AUF EINEN BLICK

- **Briefing für Claude** zum Kopieren
- **7 Schritte** von Hydrogen bis Livegang
- **Vorschau-Link** vor jedem Livegang
- **Hosting ohne Aufpreis** auf bezahlten Plänen

Das Briefing sagt Claude, was er in deinem Shop darf

Das Briefing ist eine Textdatei im Projektordner. Claude Code liest sie zu Beginn jeder Sitzung. Hier steht, was er darf, wie dein Shop aussieht und wie er Änderungen ausliefert.

Die Hydrogen-Vorlage bringt `CLAUDE.md` und `AGENTS.md` schon mit, die eine verweist auf die andere. Häng das Briefing unten an `AGENTS.md` an und ersetz die Angaben in eckigen Klammern durch deine. Anthropic empfiehlt weniger als 200 Zeilen pro Datei, weil längere schlechter befolgt werden. Das Briefing hat 47.

Briefing für Claude

```
## Briefing für Claude: [Name deines Shops]

### Worum es geht
Das ist der Shop von [Firma] unter [shop-domain.de]. Er läuft auf Shopify Hydrogen (React
Router) und wird bei Shopify Oxygen gehostet. Du baust hier Funktionen, für die wir sonst
Apps oder Freelancer bezahlen.
Für alle Fragen zu Shopify-APIs nutzt du das Shopify AI Toolkit.

### So sieht der Shop aus
- Zielgruppe: [wer bei uns kauft, in einem Satz]
- Sprache und Ton: Deutsch, [du oder Sie], [locker oder sachlich]
- Farben: [Hex-Werte], Schrift: [Name]
- Farben, Schriften und Abstände stehen in app/tokens.css. Neue Werte kommen dorthin, nie
direkt in eine Komponente.
- Prüfe jede Änderung bei 390 px (Handy) und bei 1440 px (Rechner) Breite.

### Wo was liegt
- Seiten: app/routes/, eingetragen in app/routes.ts
- Storefront-API-Client: app/lib/storefront.ts
- Warenkorb: app/lib/cart-handlers.ts, app/lib/cart.ts, app/components/CartDrawer.tsx
- Shop-Einstellungen: app/lib/shop.ts
- Styles: app/tokens.css und app/app.css
- Befehle stehen in package.json. Nutze nur die, die es dort gibt, zum Beispiel dev,
typecheck und build.

### Das darfst du
- Code in diesem Projekt lesen und ändern: Seiten, Sektionen und Komponenten bauen.
- Den Dev-Server starten und die Seite selbst ansehen.
- Rabatte und Aktionen über Shopifys eigene Rabatte laufen lassen. Schreib keine eigene
Preislogik.
- Bestellungen lesen, aber nie ändern.

### Das darfst du nicht
- Nie auf main pushen und nie einen Pull Request mergen.
- Keine Schlüssel, Tokens oder .env in Git. Werte aus .env nie ausgeben und nie in Logs
schreiben.
- Keine Produkte, Preise, Bestände oder Bestellungen im Shop ändern. Keine Kundendaten
kopieren.
- Den Checkout nicht umbauen. Er läuft bei Shopify.
- Bei Screenshots von fremden Shops baust du Anordnung und Idee nach. Bilder, Texte, Logos
```

```
und Markennamen übernimmst du nicht.
- Keine neuen Pakete installieren, ohne vorher zu fragen.

### So lieferst du Änderungen aus
1. Lege für jede Aufgabe einen eigenen Branch an (claude/<thema>).
2. Sag mir vor einer größeren Änderung in drei Sätzen, was du baust, und warte auf mein
Okay.
3. Danach müssen npm run typecheck und npm run build ohne Fehler laufen. Sieh dir die Seite
im Dev-Server an.
4. Committe und pushe den Branch. Oxygen baut bei jedem Push ein Vorschau-Deployment. Gib
mir den Link dazu.
5. Ich schaue mir den Link an. Erst nach meiner Freigabe wird in main gemerged, und erst
dann geht die Änderung live.

### Geheimnisse und Zugang
- PRIVATE_STOREFRONT_API_TOKEN, SESSION_SECRET und der Admin-Schlüssel liegen nur in .env
und in den Umgebungsvariablen bei Oxygen.
- Prüfe vor jedem Commit, dass .env in .gitignore steht.
- Für Bestellungen holst du dir mit SHOPIFY_CLIENT_ID und SHOPIFY_CLIENT_SECRET aus .env
einen Token (Client Credentials Grant, POST
https://[shop].myshopify.com/admin/oauth/access_token). Er gilt 24 Stunden, danach holst du
einen neuen.
- Fehlt dir ein Schlüssel, frag mich. Erfinde keinen.
```

Richte Shop, GitHub und Hosting in drei Schritten ein

1 Sichte deine Apps und entscheide, was Claude nachbauen soll

Öffne im Admin **Einstellungen**, dann **Apps**. Schreib zu jeder App auf, was sie macht und was sie dich im Monat kostet.

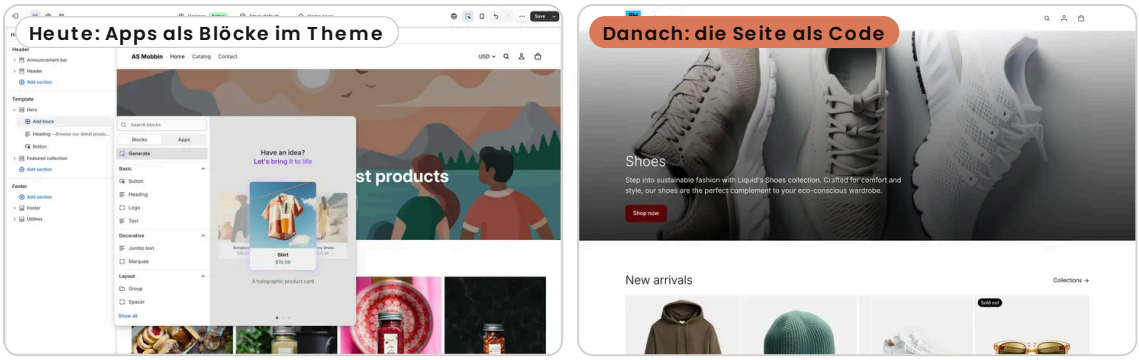
Gut nachbauen lässt Claude: Bundles, Gratis-Beigaben, Upsells, Bewertungsanzeigen und Sektionen, die du bei anderen Shops siehst. Bei Apps für Zahlung, Abos oder B2B-Preise rechnest du mit mehr Aufwand. Die behältst du am besten, bis du sicher bist.

AUS DER PRAXIS

Bei uns läuft keine App mehr

Wir haben einen Shop mit 50.000 € Umsatz komplett von Claude Code umbauen lassen. Heute läuft dort keine einzige App mehr, und er verkauft mehr als vorher.

Hydrogen ersetzt dein Theme. Die Seite besteht dann aus Code in deinem GitHub-Repository, und Apps, die sich als Block ins Theme einklinken, kommen dort nicht von allein an. Deshalb baut Claude ihre Funktion neu.



Links: Mobbin, Shopify-Admin. Rechts: Hydrogen-Vorlage mit Demodaten.

2 Leg den Storefront an: Shopify erstellt Repository und Hosting

Öffne die Vorlage mit Deploy to Oxygen. Dein Shopify-Admin öffnet die Hydrogen-Seite, falls nötig installierst du den Hydrogen-Kanal. Das Repository entsteht in deinem GitHub-Konto, du brauchst also eins.

1. Gib dem Storefront einen Namen.
2. Lass **Set up GitHub continuous deployment now** an und klick auf **Connect GitHub**.
3. Wähl dein Konto und leg ein **neues** Repository an.
4. Klick auf **Create**. Unter **View GitHub Actions** siehst du den Build.

Shopify legt das Repository mit der Vorlage und einem GitHub-Workflow an, der bei jedem Push baut und auf Oxygen ausliefert. Die Adresse ist zuerst privat, nur eingeloggte Mitarbeitende deines Shops öffnen sie.

Oxygen kostet auf bezahlten Plänen nichts extra (Stand Oktober 2026), auf Agentic-Plänen gibt es Oxygen nicht. Entwicklungs- und Testshops haben keine öffentliche Umgebung.

3 Hol dir die Kopie auf deinen Rechner und starte sie

Installiere Node.js 22 oder 24 auf deinem Rechner. Dann klonst du das neue Repository, verbindest es mit deinem Storefront und startest den Shop:

Shop lokal starten

```
git clone <Adresse deines Repositorys> mein-shop
cd mein-shop
npm install
npx shopify hydrogen link
npx shopify hydrogen env pull
npm run dev
```

Bei `link` wählst du den Storefront aus, den du eben angelegt hast. `env pull` schreibt die Variablen deines Storefronts in `.env`. Danach läuft der Shop unter `localhost:5173`.

Meldet `env pull`, dass ein geheimer Wert fehlt, trägst du ihn selbst in `.env` ein. Die Datei steht in der `.gitignore` der Vorlage und landet nicht im Repository. Ohne privaten Storefront-Token zeigt die Vorlage Demodaten von mock.shop.

Gib Claude Briefing und Zugang, aber nur so viel wie nötig

4 Starte Claude Code im Projektordner und gib ihm das Shopify-Wissen

Öffne die Claude-App, wechsle in den Tab **Code** und starte mit **+ New session** eine Sitzung in deinem Projektordner. Die Option **worktree** gibt jeder Aufgabe eine eigene Kopie des Projekts. Starte im Modus **Plan**, dann schlägt Claude erst vor, was er ändern will.

Im Terminal bekommt er zwei Werkzeuge:

Hydrogen-Skills und Shopify AI Toolkit

```
npx @shopify/hydrogen@latest setup
claude plugin install shopify-ai-toolkit@claude-plugins-official
```

Der erste Befehl kopiert Hydrogens Skills nach `.claude/skills`. Der zweite installiert das Shopify AI Toolkit mit Doku, API-Schemas und einer Prüfung für GraphQL. Sind die Skills veraltet, meldet `npm run dev` das und nennt den Befehl zum Aktualisieren. Zum Schluss hängst du das Briefing an `AGENTS.md` an.

5 Gib Claude einen eigenen Zugang mit genau den Rechten, die er braucht

Der Zugang ist eine App im Dev Dashboard von Shopify. Du legst sie an, vergibst die Rechte und installierst sie in deinem Shop. Gehört dir der Shop, hast du dafür alle Rechte. Mitarbeitende brauchen **App development** mit der Stufe **Develop**.

1. Öffne im Admin **Einstellungen**, **Apps** und **1** **Develop apps**.
2. Klick auf **Create app** und wähl **Start from Dev Dashboard**.
3. Trag unter **Versions** bei Scopes nur ein, was Claude braucht, etwa `read_products` und `read_orders`. Dann **Release**.
4. Klick auf **Install app**, wähl deinen Shop und bestätige mit **Install**.
5. Kopiere unter **App settings** die **Client ID** und das **Client secret** in deine `.env`.

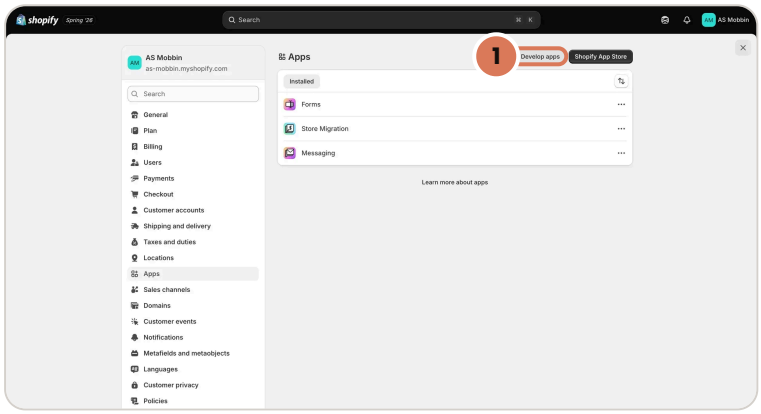


Bild: Mobbin, Shopify-Admin.

> ⚠ **ACHTUNG**
>
> **Der Schlüssel kommt nie auf GitHub**
>
> Wer das Client secret hat, hat Claudes Rechte an deinem Shop. Schreib es nur in `.env` und, fürs Hosting, als geheimen Wert in die Umgebungsvariablen deines Storefronts (**Make this value secret**). Prüf vor jedem Commit, ob `.env` in der `.gitignore` steht. Ist das Secret doch im Repository gelandet, erzeuge ein neues.

`read_orders` deckt Bestellungen der letzten 60 Tage ab. Ältere braucht `read_all_orders`, das Shopify freigeben muss. Geschützte Kundendaten wie Namen und Adressen gibt Shopify an Apps nur nach einer Freigabe heraus.

Mit Client ID und Client secret holt sich Claude einen Token, das steht im Briefing. Das klappt nur, wenn App und Shop zur selben Organisation im Dev Dashboard gehören, sonst meldet Shopify `shop_not_permitted`. Dann nimm den Weg über die Shopify CLI: `shopify store auth --store dein-shop.myshopify.com --scopes read_products,read_orders` speichert einen Token mit genau diesen Rechten.

Nichts geht ohne deine Freigabe live

6 Lass Claude auf einem Branch bauen und prüf den Vorschau-Link

Bei jedem Push baut Oxygen ein Deployment mit eigenem Vorschau-Link. Die Umgebung **Production** folgt nur dem Standard-Branch deines Repositorys, meist `main`. Alle anderen Branches landen in der Vorschau. Solange Claude auf seinem Branch arbeitet, ändert sich dein Live-Shop nicht.

Die Vorschau ist privat, nur eingeloggte Mitarbeitende deines Shops sehen sie. Ab dem Basic-Plan erzeugst du über **Share** und **Anyone with the link** einen Link für Leute ohne Login.

Zwei Prompts für den Anfang:

Sektion aus einem Screenshot

```
Im Anhang ist ein Screenshot einer Sektion aus einem anderen Shop. Baue sie auf
unserer Produktseite nach (app/routes/product.tsx).
Übernimm Anordnung und Idee. Nimm unsere Farben, unsere Schrift und unsere Texte.
Übernimm keine Bilder, Texte, Logos oder Markennamen aus dem Screenshot.
Die Sektion muss bei 390 px Breite genauso gut aussehen wie am Rechner.

Halte dich an das Briefing in AGENTS.md:
1. Sag mir zuerst in drei Sätzen, was du baust, und warte auf mein Okay.
2. Arbeite auf einem neuen Branch, nie auf main.
3. Lass npm run typecheck und npm run build laufen.
4. Pushe den Branch und gib mir den Link zum Vorschau-Deployment.
Merge nichts.
```

Bundle mit Shopify-Rabatt

```
Ich will auf der Produktseite von [Produkt A] ein Bundle anbieten: Wer [Produkt A] und
[Produkt B] zusammen kauft, bekommt [10 %] Rabatt.

Den automatischen Rabatt lege ich selbst im Shopify-Admin an. Du baust auf der
Produktseite einen Block "Zusammen kaufen" mit beiden Produkten, dem Gesamtpreis und
dem gesparten Betrag. Ein Klick legt beide Produkte in den Warenkorb.
Schreib keine eigene Preislogik. Zeigt der Warenkorb den Rabatt nicht an, sag es mir,
statt eine eigene Rechnung zu bauen.

Arbeite nach dem Briefing in AGENTS.md: Plan in drei Sätzen, neuer Branch, typecheck
und build, Vorschau-Link. Merge nichts.
```

Erst wenn du den Branch auf GitHub in `main` mergst, geht die Änderung live. Das Briefing verbietet Claude diesen Schritt.

MEIN TIPP

Hol dir die Vorschau-Links direkt in den Pull Request

Unter **Storefront settings**, **Oxygen deployments** aktivierst du **Comment on pull requests with deployment preview links**. Dann steht der Link direkt im Pull Request und du musst ihn nicht suchen.

7 Hol die alte Version zurück, wenn etwas schiefgeht

Ein Rollback zeigt die Adresse deines Storefronts auf ein früheres Deployment. Es löscht nichts und baut nichts neu.

1. Klick auf der Übersicht deines Storefronts bei der Umgebung auf **…**
2. Wähl **View deployments**.
3. Klick beim gewünschten Deployment auf **…**, dann auf **Make this the current deployment**, und bestätige.

Ein älteres Deployment behält die Umgebungsvariablen von damals. Hast du seither welche geändert, prüf vor dem Zurückholen, ob es noch läuft. Mit dem nächsten Push zeigt die Adresse wieder auf das neueste Deployment. Rollbacks gehen für die Umgebung Production und für eigene Umgebungen, nicht für die Vorschau.



Automatisierungs-Call für Agenturen, Coaches und Dienstleister

In 30 Minuten schauen wir gemeinsam, wo KI-Agenten und automatisierte Prozesse in deinem Betrieb zuerst Zeit und Geld sparen. Kostenlos.

FÜR WEN IST DAS?

- Agenturen, Coaches und Dienstleister
- ab 10.000 € Monatsumsatz
- ein klarer Flaschenhals im Tagesgeschäft
- du willst KI im Betrieb einsetzen

Call buchen

api.leadconnectorhq.com/widget/booking/mXsszCow3Sxqm4QEnOxK

Scannen & Termin wählen