



Shopify technical protocols

IFG Technical Protocol

Designing and running a Shopify ecommerce store

Francesco Guiducci

Edition October 2026 · English

ifgdigital.com

For personal or internal business use; unauthorized distribution is prohibited.

Contents

Who this manual is for	3
1. Conversion: measure the journey before changing it	4
2. Total cost: compare alternatives with the same scope	6
3. Shopify architecture on Basic, Grow and Advanced	8
4. Catalogue data: metafields, metaobjects and ownership	11
5. ERP integrations: reliable delivery and reconciliation	14
6. Migration: preserve data, journeys and operations	18
7. Performance: field measurements and laboratory diagnosis	20
8. Practical theme interventions	22
9. Apps and technical debt: decide what to retain	25
10. Design: make product value understandable	27
11. Sales channels: consistent catalogue, orders and signals	28
12. International markets: prices, languages and delivery	30
13. Analytics: reconcile events and understand margins	32
14. CRO: experiment without confusing correlation and cause	34
15. Retention: give customers a concrete reason to return	36
16. Email: sender identity, automations and content	38
17. SEO and AI search: verifiable content and accessible journeys	39
18. B2B on Basic, Grow and Advanced	41
19. AI in operations: tools, data and action control	43
20. Margins, cash and business transferability	46
Appendix A. Project records	47
Appendix B. Template and market test plan	49
Appendix C. Diagnostic exercises	50
Appendix D. Operational glossary	52
Appendix E. Sources and knowledge maintenance	54

Who this manual is for

This manual sets out a way to make decisions about a store, implement them and check their effects. The starting point may be a new project, a migration or an existing store that needs maintenance. Priorities change; reliable data, clear responsibilities and checks on actual behaviour remain necessary.

I work as a freelancer under the IFG Digital name; this manual belongs to IFG eCommerce®, the Shopify line of IFG Digital. I personally handle the design, development and optimisation of Shopify projects on Basic, Grow and Advanced. The manual covers Liquid themes, CSS and JavaScript, data structures, integrations and apps, alongside the commercial processes the code must support. Availability also depends on country, channel, app and configuration. A requirement outside this scope needs to be recognised before a solution is proposed.

You can read the chapters in order or consult them during a project. For a slow store, start with measurement in chapter 7 and then use the interventions in chapter 8. For an integration, define the data in chapter 4 before designing reliable event delivery. A migration also requires chapter 6. The commercial chapters explain how to evaluate effects without assigning results to a technical change that the evidence does not support.

Numerical examples are explicitly identified as teaching exercises. They do not describe clients or results achieved by IFG. Code fragments illustrate a bounded function: adapt them to the project's schema, API version, permissions and environment. Syntax validation is not a store acceptance test.

Use, responsibilities and updates

This material is educational. It is not a store certification, a guarantee of financial results or legal, tax or customs advice. Decisions requiring those skills should be checked with the relevant professional against the actual circumstances. Using Shopify or an app does not by itself establish that every merchant obligation has been met.

Technical sources appear in the chapters and at the end of the manual. This revision considers documentation consulted on 17 October 2026. Admin API examples use stable version 2026-07, supported until 16 July 2027; on 17 October 2026 the latest stable version is 2026-10, and the examples have not been revalidated against it. Before reusing them, check the changelog, version support and feature availability. Manual updates and store support depend on the terms actually agreed, with no implied commitment to unlimited maintenance.

Use this material under the licence through which it was supplied and applicable rights. Examples do not transfer rights to third-party themes, apps, brands or materials.

How I approach a project

Before changing a page, I want to understand its task. A product page may need to establish whether a spare part is compatible; a service page should explain the work included; a cart should let the customer review quantities and costs. Design also makes the brand recognisable, gives information a hierarchy and supports people using different devices or assistive tools.

Diagnosis starts with evidence. A reproducible JavaScript error, missing product dimensions, an absent shipping rate above a threshold and rising acquisition costs are different problems. They may produce the same symptom, fewer purchases, but require different interventions. Conversion rate therefore cannot provide an automatic diagnosis.

Every change needs an owner, a reason and a way to verify it. I save the previous configuration and prepare recovery before making the change. Theme work belongs in a test environment: changing theme code does not automatically isolate global settings, products, apps or payments. I separately check which operations affect data shared with the public store.

Finally, I leave documentation people can use. The recipient should know where to update content, which integrations are active and how to recognise a fault. A solution that depends on knowledge held only in its creator's memory needs more care whenever management changes.

1. Conversion: measure the journey before changing it

A visit may end without a purchase for many reasons: the person is comparing products, arrives from an irrelevant search, cannot find a variant or encounters an error. To identify the right intervention, separate intent, information and functionality. Treating traffic as a fluid governed by physical laws does not provide a reliable measure of human behaviour.

Define the denominator

“Conversion” is incomplete without an event and a population. You might measure sessions completing a purchase, users purchasing, quotation requests or subscriptions. These indicators are not interchangeable. Orders and converted sessions may also differ: one session can include several orders.

Create a specification for each metric before comparing reports:

Field	Decision to record
Event	Which action counts as the outcome
Denominator	Sessions, users, contacts or eligible customers
Window	Dates, time zone and reporting delay
Exclusions	Tests, internal traffic, identified bots, cancelled orders
Source	Report, export or query used
Segments	Device, country, channel, new and returning customers
Limitations	Consent, sampling, incomplete identification

A difference between platforms does not prove that one is wrong. They may count different events, use different attribution windows or receive different signals. Keep the definition beside the value; do not combine figures merely because their names match.

Read the funnel in context

Build a consistent journey: page entry, product selection, add to cart, checkout start and purchase. Specify whether the steps must happen in the same session and in that order. Open and closed funnels do not necessarily produce the same values.

Consider this teaching example based on sessions: 10,000 enter, 4,000 view at least one product page, 500 add an item, 300 start checkout and 180 purchase. The final ratio is 1.8%; completion from checkout is 60%. You cannot attribute all 9,820 sessions without a purchase to a website defect. You can investigate the product-to-cart step and check why some checkouts remain incomplete.

Segment with a specific question. If the problem affects one country, check catalogue, currency, shipping and payments. If it concerns a product family, check variants, availability and information. If it coincides with a campaign, compare the advertisement's promise with its destination. If it appears on mobile, test selectors, keyboard, errors and buttons at the affected dimensions.

Do not create so many segments that each becomes too small. Two fewer purchases in one day can move a percentage substantially. Report observation counts and compare periods with similar weekdays and commercial conditions. During a launch or promotion, record the event instead of treating it as noise.

Turn a symptom into a testable hypothesis

A useful hypothesis identifies the interruption and the proposed mechanism. "The site should look more modern" does not identify a test. "An unavailable variant remains selected and the button does not explain how to change it" lets you reproduce the case, correct the behaviour and check the result.

For each issue, retain URL, language, variant, device, consent state and steps. Add expected and actual behaviour. A short recording can show a blockage; logs help identify its cause. Redact personal data and tokens before sharing evidence.

Separate defects that prevent a task from opportunities to experiment. An incorrect price or unusable button needs correction. The presentation of a commercial offer often needs an experiment. An element is not unnecessary simply because it does not directly produce a purchase: support, accessibility, required information and consent management have other purposes.

Performance and financial results

Reducing the time needed to see or use a page can improve the experience. There is no universal formula turning 100 milliseconds into a fixed percentage of sales. Assortment, prices, campaigns and availability may change between measurements. Measure performance as described in chapter 7 and evaluate commercial effects using chapter 14.

Recalculate a conversion example with explicit assumptions. With 100,000 sessions, an orders-to-sessions ratio of 1.4% and average net revenue per order of €85, the result is 1,400 orders and €119,000 revenue. Assuming a 2.1% ratio with everything else unchanged gives 2,100 orders and €178,500. The

€59,500 difference is incremental revenue, not net profit.

Assume incremental contribution after merchandise, order handling, payments, seller-funded shipping and expected returns is 35% of that revenue. This gives €20,825 before additional project costs and other fixed costs. If incremental monthly costs are €2,000, €18,825 remains after those costs. The result depends on the conversion assumption: it does not demonstrate that a technical change will produce it.

Give the negative case equal attention. If conversion does not change, only measured benefits in operating time, avoided costs or service quality remain. If returns increase, they may absorb some additional revenue. Use scenarios and identify the assumptions with the greatest uncertainty.

Chapter worksheet. Choose a journey with a reproducible problem. Define the metric and segment; save a baseline window; collect evidence of behaviour; write the hypothesis; establish functional and financial checks. If you cannot find the fault, record “cause not identified” and the next observation needed. This avoids spending on a solution selected before diagnosis.

2. Total cost: compare alternatives with the same scope

The platform subscription is one cost item. A useful comparison includes design, content, migration, integrations, apps, maintenance, training and exit. Some items will be small for a simple store; for a catalogue with several warehouses, they may dominate the decision.

Total cost of ownership (TCO) makes alternatives comparable. It does not automatically establish which will sell more. If one proposal includes an ERP integration while another relies on manual entry, comparing quotations alone gives the second an artificial advantage.

Establish a common scope

List requirements before suppliers: products and variants, markets, languages, shipping origins, operators, imported data, automations and support. Define an acceptance test for each requirement. “Multilingual” should specify which content is translated and how updates, URLs and fallbacks are handled.

Include the merchant's work. Photography, writing, data cleaning and decisions about commercial rules take time even when absent from a development invoice. Distinguish hours that become cash outflows from internal capacity used: both matter, but should not be counted twice under different names.

A readable cost sheet

Item	One-off	Recurring	Variable	Evidence needed
Design and theme	Yes	Planned maintenance	Approved extras	Scope and included revisions
Platform	Possible setup	Subscription	Contract components	Plan, currency, billing
Apps	Installation and data	Subscription	Usage or transactions	Pricing and thresholds
Integration	Mapping and development	Hosting and support	Processing volume	Flows and service levels
Content and catalogue	Initial production	Updates	Translation	Owner and quantities
Payments	Configuration	Possible fees	Transaction and FX fees	Methods and country mix
Exit	—	Possible provision	Export and transition	Ownership and portability

State whether amounts include VAT, which currency is used and how exchange rates are treated. Check pricing and contracts when preparing the proposal: a manual is not a reliable source for fees that depend on country and commercial terms.

For each year, you can use: initial cost assigned to the year + recurring costs + expected variable costs + planned interventions. When comparing distant cash flows, a financial assessment may consider present value using an explicit discount rate. Do not subtract a hypothetical business resale value from technical TCO without explaining the estimate.

Three-year teaching example

Alternative A requires €18,000 initially and €900 monthly for the costs included in the comparison. B requires €11,000 initially and €1,250 monthly. Without cost growth, discounting or residual value, A costs €50,400 over three years; B costs €56,000. A requires €7,000 more upfront and recovers €350 monthly: simple break-even arrives after 20 months.

This comparison holds only if both monthly figures cover the same activities. If B includes ten support hours and A includes two, estimate demand and adjust the model. If the project may end after twelve months, A's higher initial cost matters differently. If variable costs rise with orders, calculate at least a low-volume and a high-volume scenario.

For simple ROI on a separate intervention, use (incremental monetary benefits – incremental cost) / incremental cost, specifying period and components. A €24,000 benefit against €10,000 cost gives 140% ROI, not 240%. The benefit-to-cost ratio is 2.4. Do not call both measures ROI.

Maintenance, risk and recovered capacity

A SaaS platform manages part of the infrastructure. Project activities remain: themes, apps, integrations, content, access and configuration. “Zero maintenance” is therefore a weak forecast. Even a native feature needs testing and people who can configure it.

Estimate incidents through specific scenarios. A missing shipping rate in one market has different exposure from a general outage. Average hourly revenue is not automatically lost revenue: some purchases happen later. For a cautious estimate, identify duration, affected traffic, recoverable purchases, contribution per order and support costs. Keep reputational damage separate when it cannot be reliably measured.

Hours freed by automation can become capacity for other work. They are cash savings only when an actual expense decreases. If someone spends four fewer hours correcting orders and uses those hours to improve the catalogue, record recovered capacity; then evaluate the new activity's result. Adding salary savings and the full value of the replacement activity without considering this relationship inflates the benefit.

Choose and document

Before deciding, highlight assumptions that could change the answer: catalogue growth, operator count, new markets, dependence on an app and maintainer availability. Test sensitivity one variable at a time. If a small change reverses the choice, prepare a contract or verification that reduces that uncertainty.

The decision record contains the selected alternative, rejected options, requirements covered, excluded costs and conditions to revisit. No universal revenue threshold or maintenance percentage replaces this analysis. Keep formulas visible so an assumption can be corrected without rebuilding the reasoning.

3. Shopify architecture on Basic, Grow and Advanced

Architecture connects the public store to the activities that operate it. It includes the theme or frontend, catalogue, checkout, payments, apps, ERP and people's processes. The Shopify plan is one choice within that system.

I start by mapping boundaries. The theme controls presentation and interactions in the online store. Checkout has its own surfaces and limits. An Admin API integration acts on authorised data; administrative credentials must not appear in the browser. Analytics observes only what its implementation and collection conditions allow.

Capability matrix

For each requirement, check plan, market, required app and permissions. On 17 October 2026 Shopify documents native B2B functions on Basic, Grow and Advanced (not on Starter or Retail); this does not mean every B2B tool is available without distinction. Chapter 18 provides the detailed matrix needed to avoid unsupported rules.

Shopify Functions are distributed through apps. Public apps containing them can work on any plan, except where individual API pages say otherwise; custom apps containing Functions require Shopify Plus. This manual does not offer that route as an IFG service. For Basic, Grow or Advanced, first check native features and compatible public apps, testing the actual rule.

For checkout changes, verify the specific surface, not just the name “Checkout Extensibility”. A thank-you page extension does not imply permission to change information, shipping and payment steps: on 17 October 2026 Shopify's help centre limits apps that customise those three pages to Shopify Plus, while apps for the thank-you and order status pages are also available on Basic. checkout.liquid is no longer supported in the three steps, and for the Thank you and Order status pages checkout.liquid and additional scripts were sunset on 28 August 2025. Do not simulate discounts or authorisation by hiding prices with CSS: the system calculating the order must apply and verify the commercial rule.

Theme or separate frontend

An Online Store 2.0 theme organises pages through JSON templates, sections and blocks. It is a useful starting point when the merchant needs to maintain content in the editor. Separating structure from content avoids requiring a developer for every change, provided section schemas offer understandable controls.

A separate frontend, for example with Hydrogen, introduces an application to design, deploy and maintain. It can address particular needs but requires attention to routing, caching, localisation, search, cart, consent, analytics and accessibility. Apps designed for themes may need a different integration. Requirements and operating capacity should support the decision, not a promise of guaranteed loading time.

Consider everyday operator tasks. Who changes a promotion? Who previews another language? Who corrects availability? If the architecture makes these tasks depend on a software release, include cost and response time. If the editor supports them, check that the operator cannot easily break heading hierarchy or publish an incomplete combination.

Contracts between components

For each connection, record data, direction, frequency, credentials, owner and failure behaviour. The ERP might own quantities and costs while Shopify owns commercial content and channel publication. A field-level ownership rule is more useful than saying everything must live in the ERP.

Connection	Technical question	Operational question
Theme → catalogue	Which fields and references does it read?	Who completes and translates them?
Shopify → ERP	How is an already received order recognised?	Who manages the error queue?
ERP → stock	Which quantity and location does it update?	When is a sale blocked?
App → messages	Which event triggers sending?	Who approves content and recipients?
Payments → accounting	Which fees and refunds are reconciled?	Who resolves differences?

A contract also covers an unavailable component. Public data may be cached for a defined duration; recent availability and yesterday's availability are not equivalent. For prices, payments and stock, define whether the process stops, asks for confirmation or uses a value with an explicit expiry.

Environments and release

Duplicating a theme copies theme files and configuration, not the entire store. Products, collections, metadata, apps and many settings remain shared. Identify changes beyond theme files and prepare an appropriate test before affecting public data.

A release should connect an identifiable version to files, changed settings and acceptance evidence. Save previous files with hashes and paths. Verify the base has not changed in the meantime: restoring an entire file from an old backup could erase another operator's recent work.

Test affected templates, active languages and devices where behaviour changes. For a product-page change, include available and sold-out variants, a product without an image, quantity, cart and transition to checkout. For content, check special characters, long text, links and headings. Publication requires an explicit decision by the store owner under the agreed process.

Service continuity

Define useful fault signals: unresolved integration errors, maximum update delay, purchase-journey failures and payment or shipping reports. A service objective must state how it is measured and who responds. Do not assign contractual availability to a store based on a general supplier statement.

Prepare a short incident procedure: containment, owner, evidence collection, correction or rollback, verification and communication. Restoring a theme does not reverse orders or ERP writes. This boundary should be clear before an incident, while there is time to organise recovery.

4. Catalogue data: metafields, metaobjects and ownership

A reliable catalogue makes clear what each value means, who updates it and where it is used. A measurement without a unit, an expired certification and an outdated translation may all be present yet commercially wrong. Quality is not the number of completed fields.

From source spreadsheet to data dictionary

Before creating fields, collect a sample containing exceptions. Include simple and complex products, variants, missing values, long names, different units and discontinued items. Designing around one ideal product pushes exceptions into free-text notes that are difficult to use.

For each attribute, record name, meaning, type, unit, allowed values, required status, source, owner and destinations. Distinguish missing from zero, false and “not applicable”. Zero length should not be the fallback for an unknown length. If you use codes, maintain a table explaining them.

Teaching attribute	Possible model	Check
Product width	Dimension type	Consistent unit and plausible value
Composition	Text or defined structure	Technical source and consistent percentages
Designer	Metaobject reference	Existing entity and intended publication
Manual	File reference	Language, version and accessibility
Compatibility	References or controlled codes	Relationship review
Certification	Entity with document and expiry	Authentic evidence, scope and validity

These are design choices, not mandatory Shopify fields. Define only what the catalogue needs and someone can maintain.

Choose the container

A metafield extends an existing resource. It suits an attribute of a product, variant or another supported resource. A metaobject describes a reusable structured entity, such as a material or designer. Products can reference that entity instead of duplicating its description.

Reuse reduces some inconsistencies but widens the effect of a change. If two hundred products reference one material, editing it changes what those products display. Check references before publication. If composition changes only for a new series, a new material version may be needed instead of overwriting the historical one.

Keep identity stable: Shopify GID, source identifier and handle serve different purposes. A readable handle can change and may not be a suitable integration key. Validate SKUs against the merchant's rules; do not assume uniqueness or immutability when the catalogue does not guarantee them.

A PIM or ERP may remain necessary for editorial workflows, external channels, documents and approval processes. Metaobjects do not automatically replace those functions. Compare required operations and management cost as described in chapter 2.

Size and performance

Shopify documents up to one million entries per metaobject definition; definitions are capped at 256 per app and 128 per store on Basic, Grow and Advanced. Storage capacity is not permission to render thousands of entries on every page. Load and display only what the user's task needs.

For writes to JSON-type fields from API 2026-04, the limit is 128 KB (64 KB for most other types). Apps that used these fields before 1 April 2026 stay at the 2 MB limit, and values above 128 KB remain readable. Check applicability before planning a migration. Size is measured in bytes, not simply characters.

If a value is too large, inspect its model first: does it contain encoded images, history, duplication or unused data? Splitting JSON without a reconstruction contract moves the problem into code. For large documents, consider file references; for independent entities, metaobjects; for internal system data, retain the reference the store actually needs.

Rendering time also depends on references, loops, generated HTML and browser resources. No structure alone guarantees LCP or TTFB. Compare before and after, including absent or unpublished objects.

Access and publication

Decide which data may appear publicly. Purchase costs, customer notes, tokens and confidential information must not be exposed merely because a theme or query makes them convenient to read. Verify access configuration and publication capabilities alongside consuming code.

A field not displayed on the page may still appear in HTML, JSON or network responses. Inspect source and responses as well as screenshots. Conversely, a field available in Admin may not be accessible to the frontend surface: handle missing data explicitly.

Controlled metafield reading and writing

Admin GraphQL 2026-07 example for a product metafield named `custom.material`, assumed to be `single_line_text_field`. Do not execute against the public catalogue before verifying ID, definition and authorisation. The query needs product read access.

```

query ProductMaterial($id: ID!) {
  product(id: $id) {
    id
    title
    metafield(namespace: "custom", key: "material") {
      id
      type
      value
      compareDigest
    }
  }
}

```

The response lets you check the resource and read the current value's digest. For an update, use `metafieldsSet` with that digest. Writing requires the appropriate resource permissions; do not infer field type solely from its text.

```

mutation SetMaterial($metafields: [MetafieldsSetInput!]) {
  metafieldsSet(metafields: $metafields) {
    metafields {
      id
      namespace
      key
      value
      compareDigest
    }
    userErrors {
      field
      message
      code
    }
  }
}

```

This variables object is a template: replace placeholders with values read from the test environment. The material value is illustrative and must not be assigned to a product without a technical source.

```

{
  "metafields": [
    {
      "ownerId": "<verified-product-gid>",
      "namespace": "custom",
      "key": "material",
      "type": "single_line_text_field",
      "value": "Cotton",
      "compareDigest": "<digest-from-read>"
    }
  ]
}

```

For protected creation of a missing field, `compareDigest: null` differs from omitting the property: it requires the field not to exist. Resolve conflicts by rereading and comparing the new value; do not remove protection just to make the request succeed. The mutation allows 25 metafields and 10 MB total per request. Group atomicity applies to this Shopify operation, not a simultaneous ERP update.

Render according to type

This Liquid example displays simple material text and optional rich care instructions. Translation keys must exist in the theme's locale files. The `h2` assumes the page already has a main heading and this section belongs one level below it.

```
{%- assign material = product.metafields.custom.material -%}
{% assign care = product.metafields.custom.care_instructions -%}
{% if material != blank or care != blank -%}
  <section aria-labelledby="ProductDetails-{{ section.id }}">
    <h2 id="ProductDetails-{{ section.id }}">
      {{ 'products.details.heading' | t | escape }}
    </h2>
    {% if material != blank -%}
      <p>{{ material.value | escape }}</p>
    {% endif -%}
    {% if care != blank -%}
      {{ care | metafield_tag }}
    {% endif -%}
  </section>
{% endif -%}
```

Plain text is escaped. `care_instructions` is assumed to use `rich_text_field` and its dedicated renderer. Do not concatenate external HTML without a validation policy. Do not escape the entire filter output, or markup becomes text. If the block repeats within a section, add a local identifier to prevent duplicate IDs.

Test the model

Test missing fields, valid values, long text, special characters, deleted references, unpublished objects and missing translations. Absence should not create empty headings or unsupported default claims. Missing certification must not become “certified” to complete a layout.

For multiple updates, begin with a limited batch. Save original values with IDs and digests, execute the authorised batch, inspect errors and reread the same fields. Compare visible content too. Recovery must account for changes since backup: the return path also needs concurrency checks.

5. ERP integrations: reliable delivery and reconciliation

An order crosses systems with different timing and responsibilities. Shopify records the purchase, an ERP may handle accounting, a WMS prepares the parcel and a carrier reports transport events. No single transaction automatically makes all these operations indivisible.

Design assuming a call can fail, arrive twice or complete without its response being received. The goal is to reach a consistent state, know when it has not been reached and recover without duplicating orders, refunds or movements.

Ownership by field and state

Assign an authoritative source to each value. The ERP may own availability and costs; Shopify may own descriptions and publication; logistics may own tracking. Exceptions, such as urgent corrections, need a procedure that prevents a later overwrite.

Define units of work: order, line, variant, inventory item, location, shipment and refund have different identities. A variant is not its inventory record; an order line can be fulfilled across shipments. Do not use the readable order number as the only global key: include store and stable ID.

Financial, logistical and commercial states must stay distinct. A shipped parcel does not prove payment settlement; a refund does not prove physical return; an authorised return does not automatically make an item sellable. Restocking depends on inspection and the approved process.

Receive webhooks without losing work

Over HTTPS, verify the signature of the original body before interpreting it. Parsing JSON and serialising it again may change bytes and invalidate the signature. Use Shopify framework authentication where available; manual verification must also handle missing headers and invalid lengths.

This Node.js module only verifies HMAC. It is not a complete webhook server and contains no real credentials.

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

export function verifyWebhook(rawBody, signature, secret) {
  if (!Buffer.isBuffer(rawBody)) return false;
  if (typeof signature !== 'string' || !secret) return false;
  // A SHA-256 digest is 32 bytes, encoded as 44 base64 characters.
  if (!/^[A-Za-z0-9+\/]{43}=$/.test(signature)) return false;
  const received = Buffer.from(signature, 'base64');
  const expected = createHmac('sha256', secret)
    .update(rawBody)
    .digest();
  return received.length === expected.length &&
    timingSafeEqual(received, expected);
}
```

After authentication, record delivery persistently and queue the work. Recording receipt and queuing processing must be reliable: a saved receipt without queued work can lose the event. An outbox in the same database is one option; other queues need an equivalent contract.

Acknowledge promptly once work is durably acquired. Do not wait for the ERP within the webhook connection: Shopify treats any response other than 200 as an error, applies a five-second timeout (one second for the connection) and, without a successful response, retries 8 times over 4 hours. X-Shopify-

`Webhook-Id` identifies the delivery; `X-Shopify-Event-Id` correlates deliveries from the same action. Delivery deduplication does not replace business-operation idempotency.

Idempotency and out-of-order messages

Idempotency means repeating the same logical operation does not add an unwanted effect. A key may combine store, order ID and operation type. If the ERP supports a unique external key, use it and verify the constraint. A simple “does it exist?” followed by “create” can fail when two processes arrive together.

Suppose the ERP creates the order but its response is lost. On retry, you must recognise the existing order. If the destination lacks idempotency, query its external key and treat ambiguity as a reconciliation condition, not permission to create again.

An old update can arrive after a new one. Where possible, compare source versions or timestamps and reread current state before sensitive changes. Arrival time does not necessarily represent source-event order. A per-resource ordered queue reduces conflicts but does not eliminate other writers.

A useful state machine distinguishes received, validated, processing, completed, retry required and awaiting review. Store attempts, last error, external identifier and last confirmation time. Logs should support diagnosis without indiscriminately retaining personal data or sensitive payloads.

Inventory updates with concurrency protection

`inventorySetQuantities` is appropriate when the writer is authorised to establish the quantity. First clarify which quantity is being set: physical stock, availability and commitments are not synonyms. Identify inventory item and location from the actual environment, not an assumed unique SKU.

From version 2026-04, use `changeFromQuantity` (introduced in 2026-01): provide the expected current quantity to enable comparison. `compareQuantity` and `ignoreCompareQuantity` were removed. The field may appear optional in the schema but must be passed as an integer or `null`; `null` skips protection and is not the recommended default.

```
mutation SetInventory($input: InventorySetQuantitiesInput!) {
  inventorySetQuantities(input: $input) {
    inventoryAdjustmentGroup {
      id
    }
    userErrors {
      code
      field
      message
    }
  }
}
```

Teaching variables template, not ready for execution:

```

{
  "input": {
    "name": "available",
    "reason": "correction",
    "quantities": [
      {
        "inventoryItemId": "<verified-inventory-item-gid>",
        "locationId": "<verified-location-gid>",
        "quantity": 12,
        "changeFromQuantity": 5
      }
    ]
  }
}

```

This requires the quantity still to be 5 before setting it to 12. If it changed, investigate and recalculate using ownership rules. Rereading 4 and blindly setting 12 again could undo a legitimate sale. The check prevents an unnoticed overwrite; it does not decide the commercially correct quantity.

Limits, batches and retries

Admin GraphQL limits calls by calculated cost per app-store pair. Documentation specifies 100 points per second for standard, 200 for Advanced and 1,000 for Plus, with a cap of 1,000 points for a single query. Read `extensions.cost.throttleStatus` rather than scheduling a fixed interval from the plan alone. A cost below the single-query maximum does not guarantee sufficient available budget.

Respect server instructions to wait. Use increasing delays with jitter and a retry limit for transient failures. Type, permission, missing-data and business-rule errors need correction; repeating unchanged requests wastes resources. A successful HTTP response may still contain GraphQL errors or `userErrors`: check every level.

Bulk Operations avoids thousands of synchronous requests for large imports. From 2026-01, documentation allows each app up to five simultaneous bulk mutations per store and JSONL files up to 100 MB. Check supported mutations and operation-specific limits too. A completed job does not prove every row succeeded: inspect its result file, which is downloadable from a URL that expires after one week.

Maximum concurrency is not a target. Batches updating the same resources may conflict. Keep an input-to-result manifest with source key, destination ID and error; repeat only rows needing recovery whose previous effect is known.

Business Central, SAP and NetSuite

For Business Central, first assess the available connector against required flows. Read version documentation, mapping and limits; test variants, discounts, taxes, refunds and locations. An official connector is a concrete option, not a guarantee that every tax or operational rule is solved.

For SAP, the landscape depends on active modules and interfaces. Do not assume every integration converts XML to JSON or uses one middleware. Identify the service provider, contract and support responsibilities. Separate environments and credentials, with coherent test data on both sides.

For NetSuite, clarify objects, locations, accounting rules and returns before selecting a connector. A returns app may coordinate requests and documents, but authorisation, receipt, refund and restocking remain distinct events. Evaluate supplier AI functions against available use cases; do not implicitly authorise credit notes or accounting changes.

Reconciliation and acceptance

Schedule checks of items that should have crossed the system. Compare counts, IDs and states within a common window; include currency, refunds and rounding for money. Totals alone can match when two orders were swapped or one duplicated while another is missing.

Test duplicates, out-of-order events, timeout after writing, temporary ERP unavailability, revoked permissions, invalid data and concurrent stock changes. Verify who receives alerts and how processing resumes. An integration is ready when normal and agreed recovery paths have evidence, not when one test order appears in the ERP.

6. Migration: preserve data, journeys and operations

Migration means deciding what to transfer, how to transform it and how to verify it still serves its purpose. Record volume is only part of the work. Relationships, states, translations, documents, URLs and the processes using them also matter.

Initial inventory

List sources and their owners: previous platform, ERP, PIM, apps, editorial files and support systems. Save original exports read-only with date, scope, record counts and hashes. Verify files open and encoding is correct. A backup never read is not yet evidence of recoverability.

Classify data operationally: import, retain in an accessible archive or exclude under a documented decision. Fiscal and personal data require specific retention rules; do not delete history merely because the public page does not need it. Do not automatically import unnecessary logs or personal data into the new system.

The catalogue also includes images, documents, taxonomies and entity references. For customers, consider consent, language and sign-in. For orders, establish which historical states can be represented without simulating new transactions. Migration must not accidentally send notifications, charge payments or alter historical inventory.

Mapping and transformations

Create a source-to-destination field map with rule, type, fallback and test. Fallbacks need a valid commercial meaning. An unknown material remains incomplete; it should not receive a plausible generated description just to fill the page.

Case	Decision needed	Verification
Duplicate SKUs	Variant identity and source key	No ambiguous links
Multicurrency prices	Currency and tax basis	Totals and rounding
Variants without images	Agreed editorial fallback	No wrong product image
Localised URLs	Equivalent language destination	Correct redirect and content
Consent without evidence	State requiring verification	No automatic marketing enrolment
Old orders	Import or archive	No new fulfilment or notification

Build entities before those that depend on them. The dependency graph may require definitions and metaobjects before products, customers before historical links and location mapping before inventory. Products-customers-orders is not a universal sequence: import method and required references determine the order.

Run a pilot containing difficult cases. A random sample may omit the anomalies that will break launch. Include apostrophes, accents, symbols, decimal amounts, products with no available variants and every active language.

Customer access and communications

Encrypted passwords from the old platform do not automatically become Shopify credentials. Plan sign-in around the chosen account system and test communications with authorised test users. Do not promise biometric authentication because accounts use emailed codes.

Check communication templates and recipients before import. Creating customers or products may trigger automations. Record side effects during the pilot and define how to disable them within the authorised scope. Before reactivating an automation, check that it will not retrospectively process every imported record.

Preserve useful URLs

Collect URLs from crawls, sitemaps, Search Console, analytics and relevant links. Map each old URL to an equivalent destination. Preserve URLs where possible. Where they change, prepare direct redirects; do not redirect every removed page to the homepage.

Shopify imposes redirect and reserved-path constraints: it redirects only URLs that no longer load a valid page and does not allow some prefixes, such as `/cart` and `/apps`. Check that the source can redirect and the destination is public. Test HTTP status, hops, final content and language. A configured redirect in Admin does not prove it runs for the actual URL.

Keep canonical, hreflang, title and structured data consistent with the destination. A canonical pointing to the former platform can undermine publication. Do not remove valid exclusions for login, cart and internal search merely to reduce excluded-URL counts in Search Console.

Final cutover

Define the cutover point and which writes may continue. If orders and stock change until the last moment, transfer deltas and reconcile finally. Record the restart point with timestamp and identifiers; a timestamp alone may be insufficient when records share a value or system clocks differ.

Assign a person to each launch decision: temporary write freeze, final import, verification, domain change, flow reactivation and monitoring. Establish stop criteria in advance. Missing prices, rates or checkout access in an active market cannot be left to last-minute judgement under pressure.

Rollback must state what returns to the previous state. Restoring a domain or theme does not erase intervening orders. Keep new events and define how to reinsert or reconcile them. Prevent both systems from simultaneously sending emails or fulfilment requests for the same order.

Acceptance and follow-up

Check counts and relationships, then targeted samples. For a historical order, compare lines, currency, discounts, taxes and refunds under the chosen model; for a product, compare variants, media, publications and metadata. A correct total count does not establish field integrity.

Run purchase journeys in agreed markets with authorised tests and no real customer data. Check accounts, search, filters, cart, rates, payments and notifications. Record separate outcomes for interface, transaction and downstream receipt.

After launch, observe 404s, redirects, indexing, orders and integration queues. Organic search may not realign on the same day. Keep a baseline and investigate by page type: neither attribute every fluctuation to migration nor ignore it because the storefront opens.

7. Performance: field measurements and laboratory diagnosis

“Fast” may describe the first server response, content appearing or a command's response. A useful report separates these moments. Customers may see a page immediately yet be unable to use a selector, or see an image late on a page that otherwise responds well.

Core Web Vitals

Documented “good” thresholds are LCP within 2.5 seconds, INP within 200 milliseconds and CLS no greater than 0.1. Assessment uses the 75th percentile of visits, distinguishing mobile and desktop. INP has been a stable Core Web Vital since 2024 and replaced FID, which should no longer be used as a reference. An internal target may be stricter but should be labelled a project target.

LCP measures when the largest eligible visible content element is rendered. It may be an image or text; it does not establish that the entire page is ready or interactive. INP considers interaction latency until the next visual update, not the complete duration of a commercial request. CLS measures unexpected layout movement; image dimensions help, but fonts, banners and dynamic content can cause other shifts.

TTFB diagnoses the first response and is not one of the three Core Web Vitals. It includes request-journey components beyond Liquid alone. TBT is a laboratory measure of main-thread blocking during the analysed interval; it helps locate expensive JavaScript but does not replace field INP.

Field data and synthetic tests

Field data represents actual experiences across different networks, devices and behaviour. CrUX data is a 28-day rolling average and is not available for every URL: it needs enough visitors. Origin-level data does not establish identical experience on every page.

A laboratory test uses defined conditions and helps isolate causes. Record tool, version, URL, viewport, CPU/network emulation, cache state and consent. Run a small series and inspect median and spread. Comparing desktop on a fast network with throttled mobile measures more than the theme change.

A preview may add scripts or bars absent from the public store. Use it to diagnose changes and record that context. Following an authorised release, verify public behaviour too. Do not present a preview score as the store's field data.

Break down slow LCP

For an image LCP, examine four intervals: initial document response, delay before requesting the image, image transfer and delay before displaying it. Each suggests different work.

High TTFB calls for inspection of requests, redirects, caching and server rendering. A late image request may result from lazy loading, JavaScript or a CSS background discovered later. Long transfer may need a smaller or properly sized asset. An already downloaded but invisible image may be held by animations or loading classes.

Retain the waterfall and trace. Identify the LCP element in that execution: a banner or dialog may become the candidate instead of the image being optimised. Do not preload every image without understanding the problem.

Diagnose INP and stability

Reproduce demanding interactions: menus, filters, variant changes, cart, search and consent. Opening a page without interacting is not a complete responsiveness test. Look for long callbacks, serialisation, layout recalculation and repeated rendering.

If a command requires the network, show an understandable state and prevent duplicate submissions. Quick feedback does not mean confirming a purchase or stock change before the response. A button may respond immediately and then report a recoverable error; preserve the user's selection and explain the next step.

For CLS, inspect filmstrips and shifted regions. Test cold cache, delayed fonts, error messages, reviews, widgets and longer translations. Reserved space must fit plausible content; an overly short fixed height may eliminate movement by clipping text.

Liquid, browser and external services

Theme Inspector helps identify Liquid costs. Read the profile in context: a complex section may do more work because it serves a useful task. Look for avoidable repetition, unnecessary catalogue scans and nested structures that multiply work.

In the browser, separate theme, app and third-party code. For each resource, record transferred size, execution time, initiator and pages needing it. Unused JavaScript coverage from one session does not prove the code is unnecessary in other states, such as checkout or an opened menu.

A repeatable report

For each intervention, record baseline, hypothesis, file or setting, subsequent measurements and functional verification. A useful result may be “the LCP request starts earlier, transfer time is unchanged, and no statistically assessable conversion change is available”. This is more precise than assigning revenue growth to a score.

Define technical budgets around project needs: resource size, main-thread work, images per template or permitted regression. An identical absolute limit for every industry may not help. Budgets should prevent regressions and guide discussion of new features.

8. Practical theme interventions

Start with an observed cause from the previous chapter. Make a bounded change, verify it and retain the diff. Changing images, apps, fonts and layout together can obscure which intervention caused the benefit or defect.

Main image

The image visible on entry should be discoverable early. Avoid lazy loading for the LCP image and consider high priority. The Liquid filters `image_url` and `image_tag` produce URLs and attributes consistent with Shopify; choose dimensions and `sizes` from the actual layout.

This example assumes a full-width hero capped at 1,440 pixels, with an image selected in the section. It is not a rule for every theme image.

```
{%- assign hero = section.settings.image -%}
{%- if hero != blank -%}
  {{ hero
    | image_url: width: 2400
    | image_tag:
      widths: '480, 750, 1100, 1500, 2000, 2400',
      sizes: '(min-width: 1440px) 1440px, 100vw',
      loading: 'eager',
      fetchpriority: 'high',
      class: 'hero-media'
  }}
{%- endif -%}
```

```
.hero-media {  
  display: block;  
  width: 100%;  
  height: auto;  
}
```

The original needs sufficient resolution; requesting 2,400 pixels does not create missing detail. Check the downloaded image at the test DPR and viewport. If the hero occupies half the desktop width, adjust `sizes` : always declaring `100vw` may download unnecessary pixels.

For different mobile and desktop images, consider `picture` with consistent sources and media queries. Two images alternately hidden with CSS may both download. Keep `alt` text relevant; use an empty alternative for a genuinely decorative image. Do not insert geographic keywords or promises into `alt` text to influence search engines.

Content below the initial viewport

For distant images, `loading="lazy"` can reduce initial work. Consider tall devices, editor reordering and different templates. A section currently at the bottom may become first tomorrow. Document positional dependencies and test reordering.

Video, 3D models and maps can load on request when the task allows it. Provide a light preview, an accessible control and an explanation of what opens. Do not require a heavy viewer to read dimensions or features that can be text.

JavaScript and components

Load component-specific code where the component exists. Before deferring it, identify dependencies and execution order. `defer` helps compatible external scripts but does not fix code assuming a synchronous library. `async` changes ordering guarantees and should be used only for independent modules.

The Theme Editor can reload a section without a full page load. Code should avoid duplicate listeners and release resources when the component disappears. A button that works initially but sends two requests after reordering fails acceptance.

Give custom components an explicit lifecycle. This example handles one event in a teaching component; it does not implement a cart or checkout.

```

class DetailsToggle extends HTMLElement {
  connectedCallback() {
    this.controller?.abort();
    this.controller = new AbortController();
    const button = this.querySelector('button');
    const panel = this.querySelector('[data-panel]');
    if (!button || !panel) return;
    button.setAttribute('aria-expanded', String(!panel.hidden));
    button.addEventListener('click', () => {
      panel.hidden = !panel.hidden;
      button.setAttribute('aria-expanded', String(!panel.hidden));
    }, { signal: this.controller.signal });
  }

  disconnectedCallback() {
    this.controller?.abort();
  }
}

if (!customElements.get('details-toggle')) {
  customElements.define('details-toggle', DetailsToggle);
}

```

For a simple disclosure, consider native `details` and `summary` first. If using the component, connect button and panel with a unique ID, use `type="button"` inside a form and preserve access without JavaScript. Performance should not be achieved by making content unreachable.

CSS, fonts and animation

Identify render-blocking stylesheets and confirm their need. Do not remove rules solely because the homepage does not use them: templates, open states, languages and breakpoints may depend on them. Critical CSS extraction must remain consistent with the rest of the theme; uncontrolled duplication creates divergence.

Limit font variants and weights to project needs. Test fallback rendering and whether the final font shifts headings or buttons. A system font avoids downloading, but the choice must respect the visual design. Check availability and licensing for external fonts.

Avoid keeping the main image invisible while an animation waits. For secondary effects, use motion that does not obstruct reading or interaction and honour `prefers-reduced-motion`. Check focus and clicks after animations finish; an intermediate screenshot does not prove the stable state.

Speculation and infrastructure

Prefetching and prerendering can improve later navigation but use network resources and may trigger behaviour before an actual visit. Check browser support, consent, analytics and excluded URLs. Do not indiscriminately preload cart, account or actions with side effects. Do not promise zero-millisecond transitions everywhere.

Early Hints and CDN configuration depend on infrastructure you control. A merchant editing a Shopify theme does not automatically control a server or external proxy. Adding intermediaries without checking compatibility, domain and cache can introduce faults. Establish that the bottleneck requires the intervention first.

Final verification

Repeat the baseline measurement series. Visually inspect crops, resolution, secondary image loading, reserved space and fonts. Test menu, variants and cart with a slow network and delayed JavaScript. Record console errors and failed requests with their initiators.

Theme Check can identify static defects and problematic practices, but passing does not establish functionality, complete accessibility or increased conversion. Combine static analysis, browser checks and measurement. If the gain is below test variability, say so; retain the change only for a verified benefit or clear maintenance reason.

9. Apps and technical debt: decide what to retain

An app may add a useful feature, store data, run a background process or load public storefront code. App count does not directly measure frontend weight. Before removing one, establish what work it performs and who depends on its data.

Dependency inventory

For each app, record name, supplier, internal owner, cost, permissions, data processed, affected pages, automations and uninstall procedure. Distinguish app embeds, app blocks, theme scripts, pixels, account extensions and external services. One function may use several surfaces.

A subscription with no recent use warrants investigation, not automatic removal. Backup, security, consent and compliance functions may be unobtrusive. Seasonal apps may produce no events for months. Ask the process owner which cases require the app and verify them.

Evidence	Possible interpretation	Next step
Script on every page	Overly broad loading	Identify where it is needed
Two equivalent purchase events	Duplicate implementation	Correlate source and event ID
App without an owner	Missing responsibility	Assign or assess retirement
Request for nonexistent resource	Residue or temporary fault	Trace the initiator
Manual-only data export	Operational dependency	Prepare procedure and format

Evaluate replacement

Compare the current function and alternative on actual cases: variants, markets, languages, combined discounts, subscriptions, returns and accessibility. A native solution may reduce dependencies but cover only part of the existing functionality. Include migration, testing and new-code maintenance in cost.

Metafields and metaobjects suit many structured-content needs; alone they do not reproduce review moderation, subscriptions, consent or warehouse logic. Shopify Functions handles specific backend logic, not arbitrary theme JavaScript. On Basic, Grow and Advanced, assess compatible public apps and native capabilities under chapter 3's limits.

If old documentation mentions Shopify Scripts, note that they were deactivated on 30 June 2026 and that Shopify names Shopify Functions as the replacement. They are a historical reference, not a new implementation choice. Shopify Plus-only work remains outside the service scope covered here.

Controlled retirement

Prepare and read the data export. Save configuration, automations, text and theme dependencies. Check whether uninstalling ends billing and external services; not every contract ends with an Admin click.

For testing, disable the affected surface according to app documentation and observe behaviour. A theme copy does not isolate all app settings. If control is global, design a separate-environment test or an authorised window. Record what cannot be isolated rather than claiming complete evidence.

For code residue, trace references, network initiators and component names. Files without obvious textual references may be called dynamically. Inspect unassigned templates, sections, snippets, assets, settings and `custom_liquid`. Remove only material whose origin and lack of need are established within scope.

Recovery may require reinstalling, restoring data and configuration as well as the theme. If the supplier deletes data, the local backup must cover what the merchant needs to retain. Check the procedure and recovery window first.

Pixels and consent

A sandboxed web pixel is distinct from server-side integration. Sandboxing restricts some page interactions but guarantees neither zero cost nor automatic compliance. Verify events, destinations, consent and behaviour after withdrawal. Do not send from the server to bypass the user's choice.

When two implementations send the same purchase, check the destination's deduplication rules. The same ID must represent the same event under its contract. Do not randomly remove the most visible pixel: that may retain the less complete source or remove signals a legitimate integration needs.

Removal acceptance

Compare recurring cost, network requests and affected journeys. Verify saved configuration and public rendering. After removing a reviews app, inspect markup, historical content and pages with no reviews. After changing a bundle, test prices, quantities, stock and order representation.

Document the function's new owner. Technical debt decreases when a solution becomes understandable and maintainable; moving a feature from an app into undocumented code may merely change ownership.

10. Design: make product value understandable

Visual design should help customers recognise the brand, understand the product and make an informed decision. A high price needs appropriate information and evidence; a dark theme, elegant font or animation alone does not justify it.

Hierarchy and content

Define each page's main task and the questions preceding it. Clothing may require fit, composition, dimensions and returns information. A technical component needs compatibility, dimensions, material and instructions. A service needs included work, exclusions, timing and responsibilities.

Order information around the decision. Heading hierarchy describes content; styles control visual size. Changing an `h4` to an `h1` for semantics does not require enlarging it: apply a scoped rule and verify layout. Avoid hidden headings added solely to pass an automated check.

Before inventing sections, review available materials. A detail photograph, measurement diagram or compatibility table may resolve a doubt more directly than promotional copy. Do not display certifications, origin, reviews or results without evidence and permission to use them.

Layout and behaviour

Choose grid, contrast and typography for the content and identity. Modular layouts may suit editorial pages; comparable lists may better serve technical catalogues. Test long copy and translations before fixing heights or proportions. Component dimensions should not depend solely on the shortest text.

Keep actions clear: “Add to cart”, “Choose a size” and “Request a quotation” describe different outcomes. Terminology should match the process. Do not replace understandable words with ambiguous labels to appear exclusive.

A sticky control can help on a long product page but may cover content or conflict with consent and chat. Test small screens, open keyboards, zoom and error messages. Displayed price and variant must remain synchronised with the current selection.

Accessibility in the design

Check keyboard operation, focus order, accessible names, contrast, messages and resizing. Dialogs need a communicated title, a closing mechanism and correct focus handling. Colour should not be the only indication of availability or error.

Decorative icons do not need redundant descriptions; icon-only actions need understandable names. Product images need relevant alternatives. Automated testing finds only some problems: add manual navigation and, for the required level, assistive tools and specialist checks.

Do not present a passed checklist as legal certification. State the technical standard, tested sample and limitations. Keeping new content accessible requires instructions for editor users too.

Images, video and 3D

Assess each format's contribution. A 3D view can help explain volume or footprint but does not always replace dimensions, photos and instructions. Consider production cost, updates, compatibility and weight. A model representing one variant must not be shown for another without justification.

For video and motion, provide captions or alternatives where needed, accessible controls and reduced-motion support. Main content should not wait for a theatrical effect. Measure cost using chapter 8 without promising universal return-rate reductions.

Reviews and trust information

Use authentic reviews and accurately describe collection methods. Do not generate testimonials, ratings or badges to complete a section. Aggregates must match the source and stated population. A service review does not automatically review every product.

Show understandable shipping, support and return conditions where needed. If cost depends on address, explain calculation without inventing a rate. Scarcity must reflect actual availability or conditions: a timer restarting for every visit does not establish a real deadline.

Test the proposition

Ask someone representative of the audience to choose a variant, find a measurement and explain what happens after ordering. Observe without guiding. Recurring mistakes inform labels and hierarchy. One aesthetic preference does not prove conversion impact.

For commercial results, use chapter 14's hypotheses and guardrails. Alongside purchases, observe support requests, errors, returns caused by insufficient information and accessibility. A change increasing order value while making conditions less understandable must be evaluated as a whole.

11. Sales channels: consistent catalogue, orders and signals

A channel introduces another set of rules: catalogue publication, permitted content, checkout, fees, payments, fulfilment and returns. Connecting it to the store does not automatically make those conditions uniform.

Channel specification

Before activation, record supported countries and products, who concludes the sale, where payment occurs and which system sends messages. Check the current channel contract; an experience available in one country may be absent in another.

Area	Information needed
Catalogue	Published products, variants, required attributes
Prices	Currency, taxes, promotions, permitted differences
Stock	Source, locations, delay, overselling safeguards
Orders	Identifier, states, ERP import
Payments	Collection, fees, refunds, disputes
Support	Owner and contact channel
Measurement	Events, consent, attribution, deduplication

Choose a pilot product group and inspect feed errors. Catalogue approval does not prove every variant can be purchased. Check final page, price, availability and purchase flow under intended conditions.

Product identity and synchronisation

Map Shopify ID, channel ID, SKU and variant. Document transformations if the channel groups variants differently. Do not correct each platform independently without updating the source: later synchronisation may restore the error.

Define acceptable operational delay per flow. Descriptive catalogue data and stock for a one-unit item may need different frequencies. An objective such as “within five minutes in 95% of cases” needs source and confirmation timestamps, not just the last call time.

For low stock, consider a consistent cross-channel availability policy. A safety reserve can reduce unfulfillable orders but also restrict legitimate sales. Calibrate it to synchronisation delay and warehouse processes without promising that overselling is impossible.

Order, fulfilment and return

Decide which event makes an order processable. Created, paid, authorised and risk-reviewed orders are not equivalent states. The ERP needs enough information to choose the next step. Preserve the source-order relationship when another platform assigns its own internal number.

Test partial shipments, cancellations and address changes. Tracking should identify the correct shipment, not be copied across the entire order indiscriminately. If warehouse processing prevents a change, communicate the limit and retain an intervention procedure.

For returns, separate request, approval, label, transit, receipt, inspection and refund. Do not restore sellable stock at carrier pickup without an explicitly justified rule. Verify the refund is recorded once and propagated to channel, accounting and reports.

Automations and ownership

Shopify Flow connects available events, conditions and actions. For advertising-platform or ERP actions, verify connector, permissions and failure handling. Do not assume a stock event can instantly pause a campaign without a specific integration.

Every automation needs an owner and an exception path. Define threshold, required data, action, limits, exit and alert. If a rule prioritises shipping, check it does not promise timing inconsistent with logistics terms. Customer classification does not itself change the purchased service.

Advertising events

A purchase may be observed by browser and server. They have different limits and requirements and do not create complete measurement of all behaviour. Respect consent and applicable rules in both implementations.

Check event name, ID, value, currency and sending time. Where deduplication exists, use a consistent contract across sources. Match-quality scores belong to the platform defining them; do not apply a Meta threshold to Google automatically. A high match score establishes neither causal attribution nor every amount's correctness.

Channel economics

Calculate net revenue, merchandise, fees, handling, shipping, returns and advertising with explicit definitions. Do not sum attributed revenue from several platforms as if it represented separate orders. Reconciliation starts from orders; attribution models interpret the journey.

Before expanding the pilot, require evidence of publication, purchase, fulfilment, refund and reconciliation. If revenue exists but costs are unavailable, label the result provisional. A new order source is sustainable when the process handles it and margins can be assessed.

12. International markets: prices, languages and delivery

Selling in another country requires an understandable offer and an executable process. Translating the menu is one task; catalogue, currency, payments, rates, taxes, documents and support also need verification.

One specification per market

Record included countries, languages, domain or subfolders, displayed currency, charged currency and payout currency. Add published products, shipping origins, timing, restrictions and responsibility for tax and customs checks. A geographical group does not necessarily share one legal regime.

Shopify requires configured shipping rates for countries activated in Markets. Multicurrency payment processing requires Shopify Payments or Adyen under documented conditions; with any other processor, customers pay in the store's default currency. Displaying a converted price does not prove checkout charges that currency.

Check each local method against merchant country, customer country and currency. Do not promise iDEAL or BLIK simply because the market exists: iDEAL is available to customers in the Netherlands and BLIK to customers in Poland, some methods are in early access, and conflicting gateways must be deactivated.

Price, exchange and revenue

Separate customer-facing price, accounting revenue and bank payout. Fees, conversion, refunds and settlement timing can cause differences. Retain original currency and amount alongside converted values and the conversion rule.

For a fixed local price, verify margin after logistics, taxes and market fees. Commercial rounding can change percentage margin. For free-shipping thresholds, assess behaviour after discounts and in each currency.

GraphQL aliases are not contextual pricing. An alias changes a returned field's name; market context must be supplied through the API surface or theme's supported mechanism. Use actual request localisation and verify checkout, which may recalculate from the address.

Translations and URLs

Inventory products, collections, pages, articles, navigation, metadata, sections, app blocks, emails and accounts. Translating Shopify content does not automatically cover every app string. After a source-language change, check whether translation is outdated and which current digest registration requires.

The language selector should lead to an equivalent page where one exists. Check public URLs, canonical and hreflang for consistency. A link adding the language prefix twice creates a different destination. For unavailable content, choose an understandable fallback without inventing a translation.

Test long text, decimal separators, date formats and units. Keep commercial conditions semantically aligned: translation must not turn indicative timing into a guarantee or an example into an included service.

Rates and thresholds

A rate depends on profile, zone, origin, weight or price and possibly app rules. For each relevant combination, test below, exactly at and above thresholds. Repeat with discounts, another profile's product, unshippable goods and multiple origins where applicable.

Teaching example: free delivery begins at €80. If paid delivery ends at €80 and free delivery starts there, boundary behaviour must match the stated rule. If premium delivery should remain purchasable at every value, a maximum order price of €80 can hide it above the threshold. Check the maximum's commercial purpose before removal: not every limit is an error.

Third-party carrier rates have plan and activation requirements to verify on the store. Decide behaviour if the external service fails. An overly cheap fallback may lose money; no fallback may prevent ordering. Document and test the choice.

Duties, taxes and contractual roles

Explain who pays what and when. DDP and other delivery terms allocate contractual responsibilities; they are not simply checkout display options. Before offering an import-inclusive price, verify service, destination, classification, origin and the party handling importation.

HS code and origin country must be accurate for the product. Automation can suggest classification but needs competent review. Warehouse location does not establish origin. Product restrictions and documentation may differ even when a shipping rate is available.

Managed Markets has geographical eligibility requirements: current documentation covers merchants based, and fulfilling, in the continental United States, Canada or the United Kingdom, and requires Shopify Payments. It does not list Italy as a merchant base country, so do not assume it is available to an Italian merchant. Even where a service acts as merchant of record for covered transactions, review the contract and residual responsibilities; this is not general immunity from tax or legal obligations.

Do not freeze tax rates, customs thresholds or fees in a manual as permanent values. Collect current sources and competent advice for the actual launch flow. Software configuration should reflect that decision, not replace it.

Market acceptance

Test country and language selection, product, variant, cart and checkout with an authorised, plausible test address. Delivery address may change conditions; a VPN alone does not verify the market. Record amounts before and after, currency, available method, rate and messages.

For an authorised test order, follow data through ERP and customer communication. Also verify refunds and contractually expected exchange differences. Do not complete real purchases or shipments merely to see whether they work without an appropriate plan and authorisation.

Finish with a market acceptance record: verified functions, exceptions, sources and next check. Where a destination is not yet served, align publication, rates and communication. A flag in a selector does not establish that the store can deliver there.

13. Analytics: reconcile events and understand margins

A report should answer a question. “How is the store doing?” is too broad for one percentage. The owner may need to know whether orders generate contribution; marketing which segment responds; operations where delays accumulate. Shared definitions and views suited to those decisions are necessary.

Separate the sources

Orders describe recorded transactions; browsing events describe observed behaviour; attribution assigns credit using a model. Their coverage differs. An order can exist without a complete browser event and can be claimed by several advertising platforms.

Choose the source for each question. For order count and status, start with the commercial system; for merchandise cost, use its authoritative system; for advertising expenditure, use statements; for receipts and fees, use available financial records. Do not force all reports to show identical numbers by removing legitimate differences.

A metric dictionary records formula, grain, recognition date, currency, exclusions and delay. A table with one row per order cannot be joined carelessly to several refund rows: the join can multiply revenue. Aggregate each dataset to the intended grain before combining them.

An explicit economic waterfall

For management reporting, start with revenue net of discounts, returns and indirect taxes, using the agreed definition. Subtract merchandise cost and variable order costs: preparation, payment, shipping paid by the seller and other relevant charges. The result is a contribution measure defined by this model.

If you then subtract advertising and other incremental costs, give the result a name that describes its scope. It is not automatically net profit: fixed staff, overheads, depreciation, interest and taxes may remain excluded. A later refund can change the period or cohort result according to the chosen method.

ROAS is attributed revenue divided by advertising expenditure. It helps interpret an attribution model when its limits are clear. A measure called POAS needs an explicit definition of the profit or contribution in its numerator. In this manual, when used, it means contribution before advertising divided by advertising expenditure; a value above 1 covers advertising within that scope, not every business cost.

Teaching example

An order generates €100 of net revenue. Merchandise costs €40, preparation and shipping €12, payment €3 and expected returns handling €5. Contribution before advertising is €40. If you allocate €20 of advertising expenditure, €20 remains after that cost. ROAS would be 5 using €100 of attributed revenue; POAS as defined here would be 2.

If merchandise cost were €65, with the other assumptions unchanged, contribution before advertising would fall to €15 and after advertising to –€5. ROAS would remain 5. This explains why margin matters without making ROAS a useless measure.

If returns cost is already included in net revenue or another item, do not subtract it twice. State whether the €5 covers only handling and transport or also lost merchandise value. Align accounting definitions with the person responsible for the accounts.

Event quality

Verify events through an authorised test journey: name, ID, timestamp, currency, value, source and consent. Check duplicates, retries, navigation and refreshes. A purchase event must not become a new sale every time someone revisits the order confirmation.

Server-side transmission does not make collection lawful if it was not lawful in the browser, and it does not necessarily reconstruct the whole journey. Shopify web pixels operate in sandboxes and follow Customer Privacy API consent signals under their extension contract; the actual implementations still need verification.

Do not treat missing data as zero. A fee that has not yet been reconciled should be marked provisional. When the report is updated, preserve the update date and recalculation rule. Do not compare a closed period with an incomplete one without saying so.

Dashboards and data architecture

A controlled spreadsheet can support a simple process. A warehouse becomes useful when sources, volumes and reconstruction needs justify it. Snowflake, dbt or other tools are not mandatory above a fixed revenue threshold.

For a pipeline, retain raw data with suitable access and retention; normalise types, time zones and keys in an intermediate layer; build final metrics on documented models. Check key uniqueness, join completeness and reconciled totals. Do not place personal data in dashboards accessible to people who do not need it.

A useful dashboard shows result, period, comparison and data quality. Relevant orders or events should be traceable for each anomaly within the user's permissions. A red arrow without context can prompt the wrong intervention.

Review cadence

Observe operational errors requiring rapid action daily. Read commercial performance over windows suited to volume. Reconcile periodically against financial records, separating recognised revenue from cash movements. A bank balance is not the period's profit: payment timing, inventory purchases, taxes and other flows affect it.

Record the decision and expected result when changing prices, budgets or configuration. The next report should answer the same question. If conditions have changed, describe them before assigning credit or blame to the intervention.

14. CRO: experiment without confusing correlation and cause

Conversion rate optimisation improves a journey through diagnosis, changes and measurement. Its value depends on the hypothesis and test. An increase after a redesign does not establish that the redesign caused it.

Fix or experiment

A broken button should be repaired and tested. There is no need to preserve the fault for a group of users merely to measure its effect. For an uncertain proposal, such as a different information order or commercial threshold, an experiment can help decide.

Write down the observed problem, affected users, proposed change, expected mechanism, primary metric and possible negative consequences. "A sticky bar will increase sales" lacks context. "People reading measurements must return to the top to select a size; a synchronised bar could reduce that step" is testable.

Do not use generic benchmarks as proof that a store is broken. Category, price, channel, country and intent change behaviour. The most useful comparison is often internal, between comparable groups or documented periods.

Test design

Define the assignment unit, minimum duration, effect you want to detect and decision rule beforehand. If assignment is by user, keep the experience consistent across subsequent visits as far as the measurement permits. Check that the tool does not introduce flicker, delays or interference with consent.

For an A/B test, check that groups are distributed as expected and assignment does not change with the outcome. An unexpected sample ratio can indicate a technical problem. Do not exclude inconvenient users after the test without a predeclared, documented reason.

Choose one primary metric and a few guardrails. If the primary metric is purchases per session, you might observe contribution per session, errors, returns and interaction times. Increasing conversion through heavy discounts can worsen the economic result.

Numbers and uncertainty

Teaching example: A records 200 purchases in 10,000 observations, B records 220 in 10,000. Rates are 2% and 2.2%, an absolute difference of 0.2 percentage points and a relative increase of 10%. These describe the same change in different ways.

The observed difference is not automatically conclusive. Estimate uncertainty using a method appropriate to the design. Do not stop as soon as a dashboard turns green or change the primary metric because another looks better. For material decisions, prepare the analysis with suitable statistical expertise.

With little traffic, a test may take too long to detect a small effect. Work on usability, errors and clarity using qualitative evidence without presenting it as proof of a causal sales increase. A before-and-after comparison can inform a decision if promotions, availability and other changes are disclosed.

Product page and cart

Check information that helps people choose: measurements, compatibility, availability, conditions and total cost where determinable. Test sold-out variants, missing images and invalid combinations. Buttons should communicate state and offer a clear recovery path.

For upsells and bundles, verify relevance, optional participation and correct pricing. Added items should be recognisable and removable. Test stock, discounts, quantities, returns and order representation. Do not hide costs or preselect unwanted additions to improve a metric.

Reviews must remain authentic and refer to the correct product or service. A testimonial count does not guarantee a percentage increase. The aim is to resolve doubts with verifiable information, not imitate a page that looks persuasive.

Checkout within the available scope

Check required fields, messages, addresses, methods and rates. Do not impose a universal field count: tax, shipping and payment requirements can differ. The ability to customise a surface depends on the plan and specific extension.

Test guest checkout where supported, accounts and available accelerated payment methods. Shop Pay being fast does not establish that every customer will use it or that it will replace all other methods. Keep the journey accessible and understandable for those choosing another option.

Decision and release

At the end, record result, uncertainty, anomalies and decision. A variant can be rejected, adopted or tested again. Preserve the code and configuration associated with the experiment; otherwise the version that generated the data becomes unclear.

After adoption, verify that the experiment has been removed correctly and no duplicate scripts or styles remain. Check the public journey and observe guardrails. A commercial decision can be revisited when prices, catalogue or audience change; one test does not create a permanent truth.

15. Retention: give customers a concrete reason to return

Repeat purchasing depends on product, service, usage frequency and relationship. A durable good has a different cycle from a consumable. A universal retention threshold cannot distinguish a problem from a different business model.

Cohorts and observation windows

Group customers by first purchase and observe their return within a defined window. To compare 90-day repeat purchasing, both cohorts need at least 90 days of observation. A recent cohort is not worse simply because it has not had time to return.

Define whether you count customers, orders or recurring revenue. Thirty per cent of revenue from known customers does not mean 30% of customers repurchased. Exclude tests and handle cancellations or refunds consistently. Disclose fragmented identity across channels.

RFM segmentation uses recency, frequency and monetary value. Choose thresholds from the actual distribution and product cycle. Someone who buys a kitchen does not automatically become inactive after 60 days; they may need accessories or assistance rather than another kitchen.

Service before the promotional sequence

Check delivery, instructions and support after the first order. A review request sent before the parcel arrives is mistimed. An offer during a dispute can damage the relationship. Use states and exit conditions, not only fixed delays.

For each flow define trigger, eligibility, exclusions, frequency and stop conditions. A recovery sequence should stop when the customer repurchases. Marketing consent withdrawal should propagate to the systems involved. Necessary service messages and promotional messages have different purposes; verify

the appropriate basis for their content and context.

Accounts and self-service

Shopify customer accounts use a six-digit one-time code emailed to the customer without requiring a password. Passkeys appear only through “Sign in with Shop”, when Shop Pay is active, so this is not a promise of biometrics or passkeys for everyone. Test code receipt, language, return to the store and order visibility.

Wishlists, order editing and other functions may require apps or extensions. Verify availability and the point at which a change becomes impossible. An address changed after fulfilment begins must not be presented as updated with the carrier without confirmation.

Store credit is a monetary balance; loyalty points, vouchers and discounts have different contracts. Shopify documents account, currency and subscription limits: credit can pay for the initial subscription purchase, not recurring bills, and is available only through customer accounts or Shop Pay. Check fees applicable to the store too: for stores created on or after 12 May 2025, orders paid with store credit incur third-party transaction fees on the amount paid with credit.

Subscriptions and reordering

Offer subscriptions when the frequency and service make them useful. Explain price, interval, renewal, management and cancellation. Test failed payments, unavailable products, variant changes and address updates. The commercial offer should match the actual installed portal.

For Shopify Subscriptions, the contract and portal support documented operations such as pause, resume and cancellation; customers manage their own contracts only if the merchant has activated subscription management, and a cancellation cannot be undone. Verify the complete journey and notifications in the store's languages.

A reorder reminder is lighter than a subscription and may suit variable consumption. Leave control with the customer and do not infer certain depletion from purchase date alone. An order may be a gift or cover several people.

Economic value

An acquisition cost already incurred does not make a repeat purchase free. Merchandise, logistics, payments, support, possible discounts and communication costs remain. Calculate cohort contribution over time using a consistent LTV definition.

Teaching example: a cohort of 100 customers generates 25 second orders with €30 contribution each before the reactivation campaign. That is €750. If messages, incentives and handling add €200, €550 remains within the model's scope. It cannot all be attributed to the campaign without estimating how many would have returned anyway.

Monitor unsubscribes, complaints and service quality alongside revenue. A sequence that brings purchases forward without increasing total contribution can look successful over a short window. Use windows appropriate to the product cycle and, where feasible, a control group.

16. Email: sender identity, automations and content

A useful email comes from a recognisable sender, reaches an eligible person and explains its purpose. Automation reduces repetitive work but requires maintenance of rules, data, domain and content.

Domain authentication

SPF, DKIM and DMARC are authentication and policy mechanisms, not certificates guaranteeing inbox placement. Before editing DNS, inventory every system authorised to send for the domain: Shopify, business email, support, billing and marketing platform.

Follow the records provided by the actual service. Shopify requires the CNAME records shown in Admin for authentication (add all of them, whatever their number) and a valid DMARC record, which is not added automatically. The domain must have only one DMARC record: several records make validation fail. Preserve previous values and verify propagation and received messages. Do not replace records without understanding which sending systems rely on them.

Gmail bulk-sender requirements (more than 5,000 messages a day, in force since 1 February 2024) include SPF, DKIM, DMARC and one-click unsubscribe for marketing or subscribed messages, alongside other technical requirements such as a spam rate below 0.3% in Postmaster Tools. DMARC policy can start at `none` ; jumping to `reject` without an inventory and verification can block legitimate messages. Check authentication and alignment in received headers.

Deliverability and list quality

Observe bounces, spam complaints, unsubscribes and rejected messages by recipient domain. Open rates can be affected by privacy protections and image preloading. Do not automatically delete contacts solely because an open was not recorded.

Define suppression segments and reactivation criteria. A suppression record may need to be retained to respect a choice or prevent further sending; this does not always mean deleting every record. Verify retention purpose and duration against applicable rules.

Do not purchase or reconstruct recipient lists without an appropriate basis. An address used for an order does not automatically authorise every campaign. WhatsApp and other channels also require their own conditions and preferences; they are not universal email replacements.

Design a sequence

For every automation, document starting event, audience, exclusions, delay, message and exit. A welcome sequence can explain a product or help someone choose; it does not need a fixed email count. Cart recovery should stop after purchase and respect sending conditions.

For post-purchase messages check delivery, product and language. Instructions should link to the correct current document. For review requests, exclude cancelled orders or unresolved problems according to the agreed process.

A win-back flow can be tested on a limited segment with a group excluded from treatment. Establish a frequency limit shared across automations: otherwise one person may receive welcome, cart, promotion and reorder messages on the same day, each valid in isolation.

Writing and layout

The subject should describe the message without pretending to be a personal reply or inventing urgency. Opening text explains why you are writing. Use one primary action for a single-purpose message while keeping necessary information, contact and unsubscribe visible.

Do not put all content in an image. Test with images disabled, enlarged text and dark mode. Email clients differ from browsers: do not assume AVIF, CSS or theme interactions work identically. Use formats and layouts tested on clients relevant to the audience.

Check prices, currency, discount codes, conditions and expiry. A cart link must not expose credentials or enable account access through an unintended shareable token. Sensitive information requires an authenticated journey.

Test before activation

Prepare authorised test recipients and entry and exit cases. Verify purchase before the delay, consent withdrawal, unavailable product, delivery error and language change. Read received messages, not only the editor preview.

Check sender, Reply-To, authentication headers, links, UTM parameters and destination page. An internal allowlisted test may not represent ordinary delivery; disclose that limit. Do not send a campaign to real customers merely to test a layout.

Measure contribution

Read deliveries, clicks and purchases with a declared attribution window. Platform-attributed revenue is not necessarily incremental revenue. Subtract costs and incentives when assessing contribution and consider purchases that would have happened without the message.

Keep a version log of flows. After catalogue or policy changes, search automations and templates for previous wording. An automated system continues sending what was configured: quality also depends on who maintains its content and conditions.

17. SEO and AI search: verifiable content and accessible journeys

Search visibility requires useful pages that relevant systems can access and that accurately describe their offer. The technical objective is to reduce obstacles and ambiguity. No file, schema or method guarantees indexing, ranking or citation by an assistant.

Page inventory

List templates, URLs, languages, HTTP status, canonical, indexability and internal links. Include products, collections, services, editorial pages and articles. Unassigned templates are part of the source and may contain outdated material that returns when someone uses them; inventory them separately from public pages.

Define each page's intent and main content. Related subjects do not automatically make two pages duplicates; technical URLs representing the same page may be legitimate alternatives. Distinguish useful content, technical variants, removed pages and restricted journeys.

An excluded Search Console URL is not always an error. Alternative canonicals, redirects and noindex can be intentional. Read examples and causes before trying to lower a total. Record the report date too: crawl data may predate current changes.

Robots, noindex and canonical

`robots.txt` controls crawling by crawlers that respect it. It is not data access control and does not guarantee removal from the index. If a crawler cannot read a page, it may not see its `noindex`.

This distinction does not justify indiscriminately unblocking login, account or other technical paths. Verify platform behaviour and reduce unnecessary URL generation where appropriate. An account link with `return_to` can generate many combinations; review theme-generated links and consistent use of `nofollow` without treating it as a deindexing guarantee.

Before changing robots rules, test positive and negative URLs. A rule matching `page=1` may also match `page=10` if the value is not delimited. Consider parameter order, separators and language prefixes. Keep documented examples and check the final generated file, not only its Liquid source.

A canonical signals the preferred version among duplicate or very similar content; it is not a redirect. Use consistent absolute URLs and check conflicts between theme, apps and settings. Google can select another canonical. Do not use canonical to erase pages with distinct intent.

Languages and internal navigation

Verify correspondence between localised versions, public URLs and hreflang. Each language should reference the appropriate alternatives under the chosen implementation. An English page should not declare an Italian canonical merely to remove an apparent duplicate.

Check menus, breadcrumbs, content links and app links. A manually written link can lose or duplicate the locale context. Titles, descriptions and visible copy should describe the same offer in every language. Structured data should not retain services removed from public text.

Titles and content

The title should identify the page without keyword accumulation. H1 describes the main content; other headings create a comprehensible hierarchy. If H1 is missing, correct the actual title instead of adding an invisible one. Verify that the semantic change does not accidentally alter sizing or layout.

Write content answering concrete questions: included work, compatibility, dimensions, costs, timing, limits and sources. For an individual professional, use their actual identity and role. Do not turn local availability into nonexistent offices or one person's work into a structured team.

Audit the complete source for outdated claims: hidden sections, `custom_liquid`, JSON, snippets, configuration, translations, comments and text assets. Hidden text is not an appropriate way to add locations or services solely for search engines. Screen-reader content serves another purpose and should not automatically be removed as spam.

Structured data

Use properties describing real facts consistent with visible content. Identify entities through stable `@id` values and connect relationships without contradictory organisation definitions. Two JSON-LD blocks are not wrong simply because there are two; the issue may be inconsistency, needless duplication or incorrect relationships.

Before removing common markup, check template exceptions. If a custom graph excludes product and article pages, those templates may rely on theme markup. Global deduplication could remove their only remaining definition. Build a template matrix and preserve the necessary entities.

Do not invent ratings, reviews, locations, credentials or certifications. Valid syntax does not establish truth or rich-result eligibility. Compare source, content and the guidelines for the desired feature.

AI search and dedicated files

Google states that existing SEO practices apply to AI Overviews and AI Mode and that no new machine-readable files, AI text files or special markup are needed: a page must be indexed and eligible to appear with a snippet. Treat any `llms.txt` as a documentation experiment without citation promises and without replacing maintained pages, sitemaps and content.

Search crawlers, training crawlers and action-taking systems have different purposes. Set policy using documentation and merchant objectives; do not open every bot because its name contains “AI”. MCP and UCP concern interaction protocols and capabilities, not synonyms for SEO.

To observe brand representation, use repeatable questions, dates and context. One assistant response is an observation, not a complete market sample. Correct inaccurate public sources first; do not add hidden instructions addressed to models.

18. B2B on Basic, Grow and Advanced

B2B requires consistent company identity, catalogue, purchasing rules and credit management. Current Shopify documentation includes native capabilities on Basic, Grow and Advanced (not on Starter or Retail). Many wholesale journeys can therefore be designed without describing them as Plus services.

Availability and limits

On 17 October 2026, companies, locations, location permissions, quantity rules, volume pricing and payment terms are documented on all three plans. Catalogues require the new Markets and the limit is three active catalogues assigned across B2B markets in total. Contextual Markets experiences (checkout and storefront) are available on Advanced; direct company catalogue assignment and advanced payment functions such as deposits, partial payments and payment requests per fulfilment are Plus-only.

Verify store availability and documentation before quoting. “A different price list for every customer” is not equivalent to “three catalogues for market groups”. If the requirement is unsupported, redesign the process with the merchant or identify an app and supported scope without simulating authorisation in the theme.

Company, location and person

Model the legal company, purchasing locations and authorised contacts. One person may act for several locations; the selected context should determine catalogue, addresses and terms. Do not infer authorisation solely from an email domain or an uncontrolled editable tag.

Shopify distinguishes “Ordering only” and “Location admin” permissions. The former can order for assigned locations and see only its own orders; the latter sees the orders of every contact at the location and can edit its addresses. It is not automatically a hierarchical order-approval workflow. Verify the actions allowed by each role.

Prepare an access request process with necessary data checks, approval and communication. When using Shopify's dedicated form, created companies cannot order or access B2B prices until approved, under the documented behaviour; a Shopify Flow template can approve every request automatically. Test duplicate requests, existing contacts and rejection.

Catalogues and quantities

Associate products and prices with appropriate groups. Verify tier boundaries, minimum, maximum and increment. A pack multiple must agree with sales units, stock and the ERP order. Ten items and ten packs have different consequences.

Test quantities immediately below, at and above each boundary. Check discount combinations and partial availability. For restricted prices, verify they are not exposed through HTML, APIs or public caches outside the authorised context. Visually hiding a price is not access control.

For a mixed DTC/B2B store, test as visitor, consumer, company contact and a person linked to several locations. Context changes should update prices and cart according to platform rules. Do not reuse personalised price caches across different users.

Quotes and payment terms

A draft order can support assisted selling requiring checks or quotations. Define who can create, edit and send the proposal, its validity and how price or stock changes are handled. Sending the customer a link is a commercial action requiring process authorisation.

Net terms describe a due date; they do not alone provide dynamic ERP credit-limit checks. If exposure must be checked, define its source, frequency and behaviour when unavailable. Do not confuse available credit, store credit, deposits and partial payments.

Tax rules, exemptions and identifier checks need a process suited to the case. Entering a VAT number does not automatically establish validity, exemption entitlement or correct taxation. Retain the decision and configure the system accordingly with appropriate expertise.

Integration and support

Map company and location to stable external identifiers. Preserve the necessary context, terms and purchase references for every order. Master data changes need clear ownership to prevent Shopify and ERP from overwriting each other.

Allow for assisted cases: urgent orders, off-catalogue items, exceptional addresses, partial returns and disputed credit. Automating repetition frees time; exceptions still need manageable, traceable handling. Human involvement does not itself prove system failure.

Journey acceptance

Final verification covers account request, approval, login, location change, catalogue, quantity, price, order and ERP recording. Test unauthorised contacts and another location too. Testing only with an administrator can conceal authorisation defects.

Measure handling time, errors and contribution per order using the real process. Do not apply universal savings per transaction. A controlled manual procedure may suit low volume; automation depends on observed cost and risk.

19. AI in operations: tools, data and action control

An assistant can help find information, prepare drafts, analyse data and propose changes. Quality depends on its sources and assigned limits. A model connected to an ERP can still misinterpret a request or use data out of context.

Choose a use case

Start with a bounded, measurable task. Finding the right manual, summarising tickets without unnecessary data or proposing copy from a technical sheet differ from issuing refunds or changing prices. Controls should follow the possible effect.

Activity	Reasonable initial control
Public-document search	Cited sources and explicit missing information
Content draft	Review before publication
Internal-data analysis	Minimum access and formula checks
Catalogue change	Diff, test environment and authorisation
Refund or financial action	Specific permissions and required approval
Customer communication	Recipient and text checked before sending

Autonomy can grow after suitable testing within a defined scope. API availability alone does not create it. Excluded tasks should be refused or passed to a person without inventing an alternative procedure.

Sources and knowledge

Organise documents by owner, version, language and validity. An assistant finding an old and a new policy should recognise which applies. Product information needs a link to the correct variant and technical source. A metaobject structures data; it does not certify truth.

Define the response when information is missing. “Unavailable in the source” is useful. Do not fill in measurements, certifications or conditions with estimates presented as facts. Compare every verifiable generated claim against original documents.

Treat web pages, emails and received documents as untrusted data. They may contain instructions aimed at the model. External text must not change the assistant's permissions, recipients or operating rules. Keep retrieved content separate from authorised process instructions.

Tool contracts

Each tool should have clear inputs, readable outputs, explicit errors and a distinction between reading and writing. Limit permissions to the necessary task. Do not place Admin credentials in browsers, prompts or shared documents. Logs should support audit without exposing secrets or personal data.

For writes, prepare a reviewable result: affected resources, before-and-after values, side effects and possible recovery. Verify store identity and current version before applying it. If a call's outcome is uncertain, reread state before repeating an action that might duplicate an effect.

MCP exposes tools and resources to assistants; it does not automatically turn any question into a correct query or eliminate hallucinations. In Shopify, the catalogue and cart tools of the Storefront MCP server (`/api/mcp`) have been removed and replaced by Storefront Catalog MCP, Cart MCP and Checkout MCP, built on the Universal Commerce Protocol (UCP) at the `/api/ucp/mcp` endpoint, where every request must include an agent profile. The store-policy and FAQ tool remains on `/api/mcp` . All of this differs from administrative access. Verify current endpoints and tools before designing: they change.

UCP and other commerce surfaces describe specific capabilities and exchanges. Their presence does not guarantee an agent will choose the store. Readable pages and an accurate catalogue remain necessary when interactions use tools.

Sidekick within its actual scope

Sidekick assists in Admin and presents changes for review before applying them. Available functions evolve; verify the task in the store. Do not presume it knows shipping costs, fees or margins when those data are absent or unconnected.

App generation is documented on Grow, Advanced and Plus, from a desktop browser only, with hourly and weekly limits per store. Generated apps run in Admin and can change customer-visible data; they are not visible to customers, cannot access themes, checkout or account apps, and are not theme or checkout extensions. Staff need the app development permission. Test them and review their permissions.

Evaluation before use

Build cases with expected answers or behaviour: present, missing, contradictory, outdated and translated information, plus content attempting to issue instructions. For operational tools include nonexistent IDs, insufficient permissions, timeouts and already-completed actions.

Measure correctness, severe errors, review time and cost. Fluent text can hide a wrong specification; fast generation may require substantial human checking. For customer support, also assess escalation and recognition of when not to answer autonomously.

Preserve the prompt, sources and tool versions associated with the test. When a relevant component changes, repeat affected cases. Yesterday's evaluation does not establish that a new model or policy behaves identically.

Forecasting and automation

For demand forecasting, compare complex models with a simple seasonal baseline. Separate training and verification periods in time; do not use future information in predictions. Evaluate error, bias and inventory consequences rather than only an average percentage.

AI can propose reorder quantities; purchasing requires budget, minimum, lead-time and supplier-availability constraints. A statistical output does not authorise an order. Start with reviewed recommendations and expand scope only after suitable evidence.

Economics and maintenance

Count service, integration, source preparation, review, monitoring and error-handling costs. Freed hours become capacity or savings according to what actually happens, as chapter 2 explains. Do not add cash released from lower stock to operating profit as if they were the same benefit.

Define who maintains knowledge and can disable the system. A useful fallback preserves service through human contact, a form or source consultation. The intended result is a correctly completed task with less avoidable work, not maximum autonomy regardless of consequences.

20. Margins, cash and business transferability

A store's financial health requires reading revenue, costs and cash timing. More orders can consume cash if inventory grows or receipts arrive after supplier payments. Technical work must be assessed within this reality.

Distinguish the measures

Revenue is not margin; margin is not net profit; profit is not bank balance. In chapter 13's management model, contribution first deducts variable costs and then the additional costs explicitly included. For financial statements, use definitions and criteria agreed with the accounting professional.

EBITDA is a result before interest, taxes, depreciation and amortisation. It is not calculated by subtracting those items from net income. Explain and reconcile adjustments because identically named measures can have different scopes. Do not treat EBITDA as cash flow or an automatic business value.

SDE is used in some small-business valuation contexts and requires justified adjustments. It does not permit indiscriminately adding every personal expense or owner payment. Normalisation needs evidence, comparability and professional analysis of the case.

Model the intervention

If a project costs €12,000 once and €200 monthly, its first-year cost is €14,400 within the assumed scope. For a teaching example, suppose €9,600 of actually avoided costs and €8,000 of incremental commercial contribution. Total benefit would be €17,600, net benefit €3,200 and simple ROI approximately 22.2%.

If commercial contribution does not materialise, net benefit becomes –€4,800. This scenario should be as visible as the favourable one. If the €9,600 represents freed hours without reduced spending, the cash result changes again. Do not call the whole amount guaranteed profit.

In year two, do not repeat a nonrecurring initial cost, but include maintenance, changes and remaining costs. With benefits distributed over time, calculate payback from cumulative flows. Do not simply divide cost by annual benefit when flows begin late or vary substantially.

Inventory and working capital

Incorrect availability can produce unfulfillable sales or tied-up capital. Measure improvement through stock, turnover, reorder timing and customer service. Reducing inventory releases cash once; it does not generate the same revenue every year.

If better information is expected to reduce returns, measure reasons, cohort and actual cost. A one-percentage-point reduction can have different effects depending on product value, recoverability and transport. Avoid applying a universal percentage to revenue.

Procedures and continuity

Document operations that keep the store running: product creation, price changes, publication, order handling, returns, access, incidents and service renewals. Each procedure needs inputs, permissions, steps, result, exceptions and verification. Test it with someone who did not write it.

Maintain an inventory of domains, apps, licences, accounts, repositories and integrations. The owner should know ownership and recovery methods. Do not share passwords in operating documents; use appropriate access and recovery systems. Revoke unnecessary access when a collaborator changes.

Documentation reduces reliance on unshared knowledge but does not automatically determine a business's sale value. Product quality, customer concentration, contracts, margins, market and risk affect valuation. No theme work guarantees a multiple will rise from 2.5 to 4.2.

Agreed handover and maintenance

Handover links version, scope and evidence. Include relevant files and settings, content instructions, dependencies, known limits and recovery procedures. For each check distinguish static verification, preview testing and the authorised public journey.

Define who manages updates and incidents after delivery under actual support terms. Do not imply permanent monitoring. A review schedule can cover access, dependencies, integration errors, performance and content at a frequency proportionate to risk and volume.

Choose the next work

Collect open problems with evidence, consequences and estimated correction cost. Prioritise service failures, data exposure and economic errors; evaluate opportunities through scenarios and evidence. When information is missing, the next task may be a bounded measurement.

A project continues creating value when people can use it, recognise errors and update it without losing control. That is the handover I aim for: an understandable store with documented decisions and results verifiable within their actual scope.

Appendix A. Project records

Use these templates to preserve decisions and hand work to another person. Fill them with project data; empty cells identify information to collect, not permission to invent undeclared assumptions.

Initial record

Field	Value to complete
Verified store and domain	...
Plan and verification date	...
Store owner	...
Environment and theme	...
Work objective	...
Authorised operations	...
Data shared with production	...
Markets and languages	...
External dependencies	...
Acceptance criterion	...
Recovery procedure	...

Identify the theme by ID and role as well as name. Names can be duplicated or changed. Before every write, check that the environment is still the intended one and base files have not changed. The initial check does not remain valid indefinitely throughout a project.

Decision log

Date	Decision	Evidence	Alternatives	Consequences	Owner
...	...	...	...	...	...
...	...	...	...	...	...

Record why a solution was chosen and what could cause its reconsideration. Do not record every click: preserve decisions a maintainer cannot reconstruct from code, such as a fallback shipping rate or retaining a function in an app.

Change record

Field	Question
Problem	How is it reproduced?
Resources	Which files, settings or data change?
Baseline	Which version and hashes?
Change	What is the resulting behaviour?
Side effects	Which other journeys can change?
Tests	Which checks and results?
Release	Who decides, and when?
Rollback	What is restored and what needs reconciliation?

A file backup restores that file. It does not establish recovery of app-deleted data or reversal of sent messages. Describe that limit in the rollback field.

Appendix B. Template and market test plan

Build a matrix joining template, language and viewport. Choose representative URLs, including rare cases that can break rendering. Static checks remain useful for unassigned templates; runtime rendering requires a suitable authorised preview.

Area	Cases to include	Evidence
Homepage	Hero, menu, reordered sections, consent	Screenshots and interactions
Collection	Empty, many products, filters, pagination	URLs and states
Product	Variants, sold out, missing media, long text	Selection and cart
Cart	Empty, quantity, removal, error	Before and after state
Editorial page	Headings, links, translation, tables	HTML and viewport
Article	Author, images, dates, schema	Content and source
Account	Login, language, return, permissions	Authorised journey
Checkout	Address, rate, method, currency	Agreed test
Return	Request, receipt, internal handling	Separate evidence

For interactions, check stable state after animations. Verify focus, keyboard and return from dialogs. Save URL, viewport and context with screenshots; an image alone does not identify the theme or prove a button works.

Use precise internal statuses:

- **Verified:** the planned check ran and its evidence supports the result.
- **Unverified:** necessary evidence is missing; state which check is needed.
- **Blocked:** a concrete constraint prevents the check; record it.
- **Not applicable:** the case is outside scope, with a stated reason.

A skipped check has not passed. Static analysis does not become customer-journey testing. A function saved in Admin is not necessarily delivered to customers: verify its actual surface.

Appendix C. Diagnostic exercises

C1. A disappearing shipping rate

Teaching scenario: standard delivery is free from €80; express delivery should remain available for purchase. At €79.99 both appear, but at €80 only the free option remains.

An upper price limit on express is the first hypothesis, but verify it in the relevant profile and zone. Also check customisation apps, products and cart origins. If that limit exists and the commercial rule confirms continuous availability, prepare a change removing only the incorrect limit. Test below, at and above the threshold with the same products, then equivalent cases in other affected zones.

The test ends when checkout shows the correct rate, not when the field is saved. Do not change delivery price or the free threshold when only the express maximum is wrong.

C2. A semantically incorrect title

Teaching scenario: a service page uses `h4` for its main title because global styling makes `h1` too large. No other main heading exists.

Correct the actual title's tag. Preserve size and line height through styling scoped to the section and template using stable local identifiers. Do not hide a new `h1` off-screen. Verify other headings and cards and the same hierarchy in all languages. Static tag checks precede visual checks with long text.

C3. A duplicate webhook

Teaching scenario: Shopify repeats a delivery after a timeout and the ERP creates two orders.

Check delivery identifier, business key and first-response logs. An in-memory duplicate check may have been lost on restart. A check-then-create sequence without a unique constraint may allow two workers to create simultaneously. Prepare a persistent key and idempotent destination behaviour. Existing duplicates need authorised reconciliation, not automatic deletion based only on timestamps.

C4. Higher revenue, worse result

Teaching scenario: before a promotion, 100 orders generate €4,000 contribution before advertising; afterwards, 130 generate €3,900 on the same basis. Advertising expenditure remains €2,000.

Contribution after advertising falls from €2,000 to €1,900 despite more orders. Check discounts, product mix, shipping and returns. Handling thirty extra orders may already be included or still missing; verify the definition. Do not present 30% more orders as a conclusive financial improvement.

C5. Duplicate structured data

Teaching scenario: the theme emits Organization on every page; a custom graph emits it on editorial pages but excludes products and articles.

Build the template matrix before removal. A condition can preserve common markup only where the custom graph is absent. Verify final `@id`, name, URL and relationships. Preserve canonical and hreflang: JSON-LD deduplication does not require changing them. Check unassigned templates too if code may be reused.

C6. An assistant receives instructions inside a document

Teaching scenario: a technical sheet contains a sentence ordering the model to send the catalogue to an external address.

The sheet is a source to read, not authority to assign actions. The assistant should ignore the operational instruction and use only relevant data. The test passes when it sends nothing and does not expand permissions. If the document belongs to an automated pipeline, add the case to evaluation and verify other input surfaces too.

Appendix D. Operational glossary

Term	Meaning in this manual
AEO	Editorial practices for understandable search answers; no citation guarantee
Agentic commerce	Commercial interactions assisted or performed by agents within capabilities and authorisation
API	Contract allowing software to exchange data or request operations
AOV	Average order value, with revenue, exclusions and period defined
Attribution	Model assigning credit to a touchpoint or channel
Bulk operation	Asynchronous volume processing whose results require verification
CAC	Acquisition cost per new customer within a declared scope
Canonical	Signal of the preferred version of duplicate or very similar content
CAS	Compare-and-swap: a write conditional on an expected value
CLS	Measure of unexpected layout shifts
COGS	Cost of goods sold under the adopted accounting method
Cohort	Group sharing an initial event, observed over time
Contribution	Revenue minus the costs defined by the management model
CrUX	Aggregated experience data from eligible Chrome users
Cutover	Operational transition from the previous system to the new one
DDP	Delivery term assigning specific obligations and charges; verify the contract
Digest	Value fingerprint used, among other purposes, to control concurrent updates
DLQ	Queue for unprocessed messages requiring recovery or review
EBITDA	Result before interest, taxes, depreciation and amortisation; scope and adjustments need explanation
ERP	Business management system whose functions depend on modules and configuration
Eventual consistency	Consistency reached after updates propagate, not simultaneously everywhere

Term	Meaning in this manual
GEO	Name used for optimising representation in generative systems; not a replacement for SEO foundations
GID	Global identifier of a Shopify resource
HMAC	Message authentication code calculated with a secret key
Hreflang	Indication of a page's language or regional alternatives
Idempotency	Repeating a logical operation without adding unintended effects
INP	Interaction latency until the next visual update
JSON-LD	Linked-data format; assertions must remain true and relevant
JSONL	One JSON object per line, used in bulk operations
LCP	Rendering time of the largest eligible visible content element
llms.txt	Proposed file describing content to AI systems; no general use guarantee
LTV	Customer value over time; specify revenue or contribution and the window
MCP	Protocol exposing tools and resources to an AI application
Metafield	Additional data associated with a supported resource
Metaobject	Reusable structured entity defined through fields
MoR	Merchant of record for transactions covered by the contract; verify responsibilities
Noindex	Index-exclusion directive the crawler must be able to read
Outbox	Pattern recording work to send alongside the local transaction
PIM	System managing product information and workflows
POAS	Ratio of attributed profit or contribution to advertising expenditure, explicitly defined
RFM	Segmentation by recency, frequency and monetary value
ROAS	Attributed revenue divided by advertising expenditure
RUM	Real-user experience measurement
SDE	Adjusted measure used in some small-business valuation contexts
SLA	Contractual service-level commitment

Term	Meaning in this manual
SLO	Measurable service objective
SOP	Documented operating procedure
Source of truth	Authoritative source for the specified data and field
Store credit	Monetary balance used under platform and offer conditions
TBT	Laboratory measure of main-thread blocking during the measured period
TCO	Total solution cost within a stated scope and horizon
TTFB	Time to the first response byte
UCP	Universal Commerce Protocol: protocol for commerce capabilities, distinct from page crawling
Wasm	WebAssembly execution format, also used by Shopify Functions
Webhook	Event notification delivered to a destination system

Appendix E. Sources and knowledge maintenance

These sources were consulted for this revision; the technical checks date from 17 October 2026. Provider pages may change content or address. Save relevant dates and versions for a project; when a page contradicts an older note, check the changelog and the contract of the version actually used.

Shopify: data and integrations

- [API limits](#)
- [Metafield limits](#)
- [Definition and metaobject limits](#)
- [JSON limits from 2026-04](#)
- [metafieldsSet and compareDigest](#)
- [New inventorySetQuantities syntax](#)
- [Webhook verification](#)
- [Bulk imports](#)
- [Functions and availability](#)
- [Web pixels](#)

Shopify: themes and operations

- [metafield_tag](#)

- [image_tag](#)
- [Theme performance](#)
- [Theme Inspector](#)
- [LCP image animations](#)
- [Theme Check](#)
- [Migration checklist](#)
- [URL redirects](#)
- [Historical Shopify Scripts status](#)
- [Checkout Extensibility](#)

Shopify: markets, accounts and B2B

- [Markets requirements](#)
- [Local payment methods](#)
- [Managed Markets conditions](#)
- [Customer accounts](#)
- [Store credit](#)
- [Subscription contracts](#)
- [Email authentication](#)
- [B2B by plan](#)
- [Company contacts](#)
- [B2B access requests](#)

Search, performance, AI and financial measures

- [Web Vitals](#)
- [Noindex](#)
- [Canonical](#)
- [Structured data policies](#)
- [Google Search AI features](#)
- [Gmail sender guidelines](#)
- [Shopify Storefront MCP](#)
- [Sidekick](#)
- [Sidekick-generated apps](#)
- [IFRS technical paper on EBITDA](#)

Use sources during a project

Connect each source to its specific requirement, record the date and verify the environment. For APIs, pin a version, validate queries and variables, then test in the authorised context. For plans and services, check store availability too. For law, taxes, customs and accounting, provider technical documentation informs the work; it is not advice on the individual case.

Sources

- Shopify API limits: <https://shopify.dev/docs/apps/build/apis/graphql-admin/rate-limits>
- Shopify API versioning: <https://shopify.dev/docs/api/usage/versioning>
- Shopify metafield limits: <https://shopify.dev/docs/apps/build/metafields/metafield-limits>
- Shopify changelog JSON metafield size: <https://shopify.dev/changelog/reduced-metafield-value-sizes>
- Shopify inventorySetQuantities compare-and-swap: <https://shopify.dev/changelog/finalizing-compare-and-swap-redesign-for-inventory-set-quantities>
- Shopify webhook verification: <https://shopify.dev/docs/apps/build/webhooks/verify-deliveries>
- Shopify bulk operation imports: <https://shopify.dev/docs/api/usage/bulk-operations/imports>
- Shopify Functions API availability: <https://shopify.dev/docs/api/functions/latest>
- Shopify Scripts: <https://help.shopify.com/en/manual/checkout-settings/script-editor/shopify-scripts>
- Shopify checkout extensibility: <https://help.shopify.com/en/manual/checkout-settings/checkout-extensibility>
- Shopify Markets requirements: <https://help.shopify.com/en/manual/markets/getting-started/requirements-and-considerations>
- Shopify Managed Markets requirements: <https://help.shopify.com/en/manual/international/managed-markets/requirements-and-considerations>
- Shopify B2B plan features: <https://help.shopify.com/en/manual/b2b/getting-started/plan-features>
- Shopify customer accounts: <https://help.shopify.com/en/manual/customers/customer-accounts/new-customer-accounts>
- Shopify email authentication: <https://help.shopify.com/en/manual/intro-to-shopify/initial-setup/setup-your-email>
- Shopify Storefront MCP: <https://shopify.dev/docs/apps/build/storefront-mcp/servers/storefront>
- Shopify Sidekick app generation: <https://help.shopify.com/en/manual/ai-powered-tools/sidekick/generate-apps>
- web.dev, Core Web Vitals: <https://web.dev/articles/vitals>
- Google Search Central, AI features: <https://developers.google.com/search/docs/appearance/ai-features>
- Google Search Central, noindex: <https://developers.google.com/search/docs/crawling-indexing/block-indexing>
- Google Workspace Admin Help, email sender guidelines: <https://support.google.com/a/answer/81126>

Sources checked on 17 October 2026.