

COMPOSABLE COMMERCE · PLATFORM FIT

Strapi vs Sanity vs Medusa

Choose the platform around the responsibility.



GYAN SOLUTIONS

RETAIL &
ECOMMERCE

A practical architecture guide for ecommerce and digital leaders

THE DECISION LENS

The real decision is ownership, not features

The useful question is not “Which platform has the longest feature list?” It is “Which system should be trusted with this responsibility?”

Separate content responsibilities from commerce responsibilities before choosing a platform. Once ownership is clear, the integration pattern becomes easier to design, govern, and change.

LEADERSHIP TEST

One source of truth

For every important object, name the system that owns it, the systems that consume it, and the team accountable when it is wrong.

THREE JOBS · THREE CENTERS OF GRAVITY

Three platforms, three jobs

Strapi and Sanity can both sit beside Medusa. They are complementary choices when the boundary between content and commerce stays explicit.



Strapi

Structured content, modeled and managed by a team that wants clear schemas and API access.



Sanity

Flexible editorial content, rich authoring, and live-preview workflows around the experience.



Medusa

Commerce logic and transactions: carts, checkout, orders, payments, and operational workflows.

Draw the system boundary before you integrate

Make ownership visible: storefronts consume experiences, the CMS owns content, the commerce engine owns transactions, and surrounding systems remain explicit dependencies.

Channels

- Storefronts** — web · app · assisted
- Identity** — accounts · roles
- Search** — discovery · indexing

Ownership centers

- CMS layer: Strapi or Sanity**
Content · editorial experience
- Medusa** — commerce engine
Cart · checkout · orders

Operating systems

- ERP** — inventory · finance
- Payments** — authorization
- Analytics** — events · reporting

Storefronts consume CMS experiences and Medusa commerce; Medusa connects with ERP, Payments, and Analytics.

What each platform should own

Responsibility	Strapi	Sanity	Medusa	Other system
Product data	Structured catalog content	Editorial product story	Commerce product entity	ERP attributes when operational
Editorial content	Structured pages/modules	Flexible editorial experience	Consume only	—
Pricing	Reference only	Reference only	Own	ERP/pricing service input
Inventory	Display only	Display only	Consume current value	ERP/inventory system
Checkout + orders	—	—	Own	Payments authorize
Localization	Structured variants	Editorial variants	Commerce locale rules	Translation workflow

Integration patterns that stay maintainable

Choose the lightest pattern that matches the responsibility and its failure cost. Medusa documentation describes the commerce layer through API routes, workflows, modules, and a data store; its guides also describe pairing Medusa with Strapi and Sanity.

01

One-way publishing

CMS publishes content to the storefront or search index. Use APIs for reads and webhooks for controlled invalidation.

Best for: editorial pages and campaigns

02

Product enrichment

Medusa remains the commerce record while the CMS adds descriptions, media, and merchandising context.

Reconcile on a schedule.

Best for: richer product experiences

03

Event-driven workflows

Events move changes through queues and workers. Add retries, idempotency, observability, and a repair path.

Best for: high-change operational flows

Migration is a data and operating-model change

A successful migration moves more than records: it changes who models content, approves it, owns failures, and validates a live order.

1. **Inventory** — Profile records, dependencies, and quality gaps.
2. **Model content** — Define product and editorial ownership.
3. **Redirects** — Protect URLs, references, and search continuity.
4. **Integrations** — Rebuild APIs, webhooks, queues, and reconciliation.
5. **Cutover** — Run rehearsals, set freeze rules, and make a measured switch.
6. **Validate** — Test content, checkout, reporting, and rollback.

Migration work is operational work: inspect, rehearse, validate, recover.



Common architecture mistakes

Most failures are not caused by choosing the “wrong” platform. They come from leaving ownership undefined and letting convenience become architecture.



Duplicated product truth

Product data is edited in two places. Decide whether Medusa, an ERP, or a content system owns each field.



Commerce logic in the CMS

Pricing, stock, tax, and order state become editorial fields. Keep transactional rules in the commerce layer.



Editorial content trapped in commerce

Campaigns and page composition become release work. Give editors an appropriate CMS surface.



No failure owner

An integration without retries, monitoring, reconciliation, and an accountable team is not maintainable.

DECISION SCORECARD

A practical decision framework

Score the responsibility, not the brand. A platform can be a strong fit in one column and the wrong owner in another.

Criterion	Low	Medium	High	Decision prompt
Content complexity	Few types	Reusable modules	Many relationships	Who models and governs the content?
Commerce depth	Simple catalog	Multiple markets	Complex workflows	Where do transactions and rules belong?
Editorial workflow	Single author	Review stages	Live preview + teams	How much freedom do editors need?
Team skills	CMS-led	Full-stack	Platform engineering	Who will own extensions and incidents?
Integration load	Few systems	Several APIs	Events + queues	Can the team observe and repair the flow?
Migration risk	Clean data	Mixed quality	Critical legacy	What must remain reversible?

STRAPI SIGNAL

Structured content and API-led modeling are the center of gravity.

SANITY SIGNAL

Flexible editorial workflows and live preview are the center of gravity.

MEDUSA SIGNAL

Commerce logic, transactions, and workflows are the center of gravity.

Choose the Platform Around the Responsibility

Choose Strapi Development Services when structured content and API-led governance lead. Choose Sanity CMS Development Services when flexible editorial work and live preview lead. Choose Medusa.js Development Services when commerce logic and transactions lead.

Use them together when the boundary is explicit, the source of truth is named, and the failure path is owned.

NEXT CONVERSATION

Book a 30-minute Operations Fit Call

