



Cloud-Ready by 2026: Your SAP Commerce Migration Playbook

Guide built on real migrations,
not recycled advice

On July 31, 2026, SAP ends support for on-premise SAP Commerce. For any team still running legacy infrastructure, the pressure is on.

This isn't a change you can rush through or handle by copy-pasting best practices. What's ahead demands technical clarity, system-wide discipline, and a plan that accounts for every step, not just the moment you hit "go live."

This whitepaper isn't another checklist of what should happen during a cloud migration. Instead, it's built around real scenarios we've faced while moving enterprise-scale SAP Commerce platforms to the cloud. Each section draws on live migrations — the tradeoffs, edge cases, and critical moves that made the difference.

If you're looking to prep your system, your teams, and your roadmap for what cloud migration really demands — start here.

Table of Contents

From Legacy to Cloud-Native: What Changes with SAP Commerce Cloud	3
Pre-Migration Audits: What to Pay Attention To	5
What Makes SAP Commerce Cloud Migration Work in Practice	7
After Go-Live: Fixing What the Migration Didn't Solve	10
Operational Habits That Shape Cloud Success	12

From Legacy to Cloud- Native: What Changes with SAP Commerce Cloud

Probably, you've heard the value props of SAP Commerce Cloud by now: with the new setup, you leave maintenance headaches behind. You stop worrying about patching servers or sizing environments.

Instead, you get a more reliable system, faster feature rollouts, better scalability – everything needed to move faster with fewer blockers. The rest you've probably already seen on every SAP marketing slide.

But what really matters here is that you receive another type of responsibility – **pipeline discipline and runtime awareness**. That means owning how clean your releases are, and how visible your system becomes when something goes wrong.

**Sharp systems come
from sharp teams.**

Follow us on LinkedIn for grounded insights on building with clarity – from architecture to culture.

JOIN! 

These are some examples of what actually changes and what it takes to stay in control as your platform shifts.

INFRASTRUCTURE BECOMES ABSTRACTED

You no longer manage servers directly — instead, you interact with the platform through APIs and operate within the boundaries of SAP's SLAs. That means less manual tuning, but more emphasis on designing systems that work within the guardrails.

RELEASE MANAGEMENT MOVES TO THE CLOUD PORTAL

Deployments happen through one-click releases in the Cloud Portal. This simplifies delivery, but only if your CI/CD pipelines are clean and consistent.

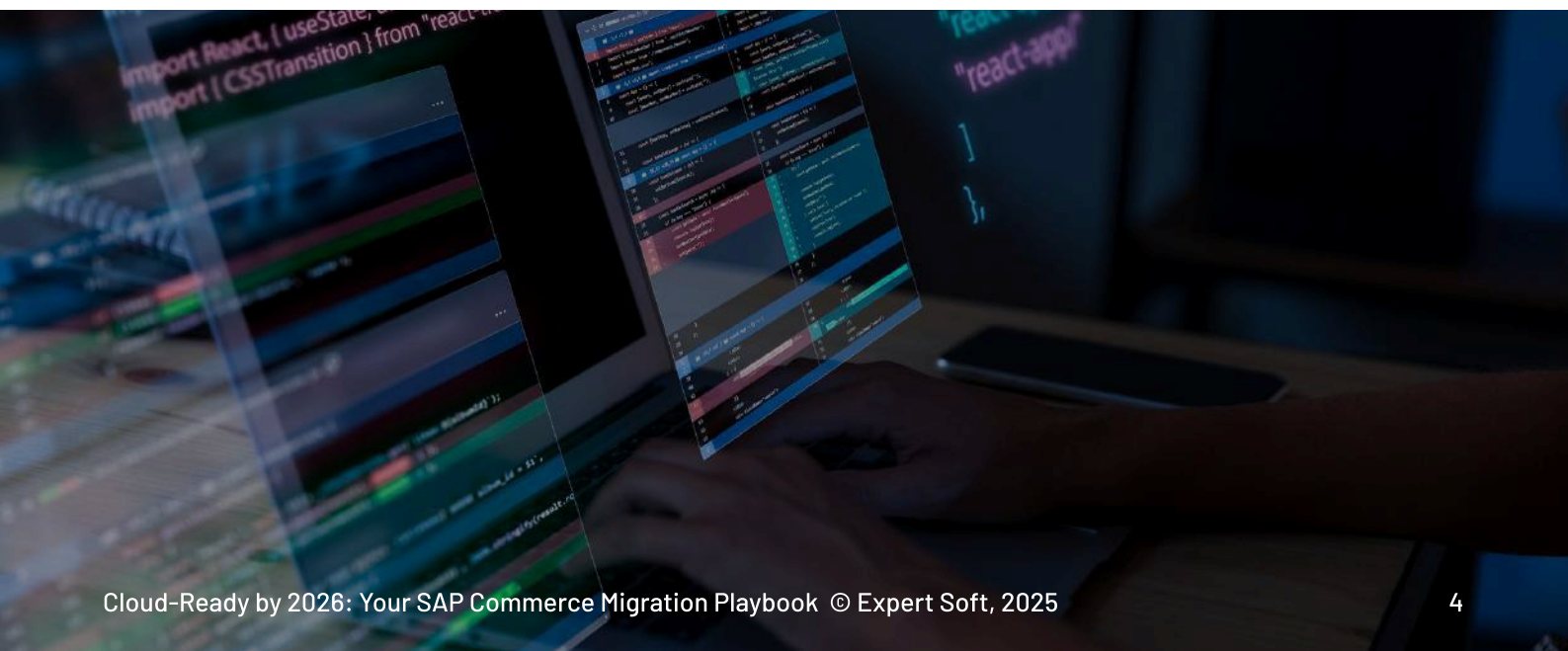
MONITORING REQUIRES NEW TOOLS

You lose traditional access to logs and system internals. To keep observability intact, you'll need to wire in tools like Kibana, Grafana, or APM early on. Without them, you're flying blind the moment something breaks.

SECURITY MODEL SHIFTS TO CLOUD-FIRST STANDARDS

You work within predefined network rules, federated auth models, and SAP's security posture, including OAuth, SAML, and integration with your enterprise IdP. This reduces low-level security overhead, but requires early coordination with your security and IAM teams.

Yes, with the new setup, you're absolutely benefiting: handing off root access to SAP means fewer headaches on your side. But you also need to adapt to new realities: clear observability and disciplined pipelines aren't optional anymore. And getting there doesn't start in the cloud — it starts with the code, logic, and jobs you bring with you.



Pre-Migration Audits: What to Pay Attention To

From our experience, a smooth migration to the cloud starts with proper preparation. That means taking a hard look at your current system and deciding what's worth bringing and what isn't. Inefficient classes and hidden vulnerabilities? Probably not the travel companions you want.

When we migrated [a Slovenian telecommunication provider to SAP Commerce Cloud](#), we made one key call early:

It took time, yes, but it paid off in stability, clarity, and fewer surprises down the line. Here're some more pre-migration best practices from that experience.

REFACTORING IS THE REAL FIRST STEP

Before we touched anything cloud-related, we focused on the code itself. Because no matter how modern the target environment is, legacy messes don't clean themselves on the way up.

- | **What we did:** started with a full audit using SonarQube. It surfaced inefficient class structures, tangled dependency cycles, and security vulnerabilities that would have made the migration brittle and error-prone.
- | **Example of inefficiency:** outdated banner system with unnecessary CMS components. We refactored it into dedicated extensions, isolating responsibilities and optimizing the build pipeline.
- | **The result:** CI/CD build time dropped from 40 minutes to 15.

Refactoring directly strengthens delivery stability. And in cloud environments, where fast, clean releases matter more than ever, that's exactly where your prep work should start.

CODE COMPLIANCE MUST BE CLOUD-AWARE

Cloud environments introduce boundaries that don't exist on-premises: you can't rely on local file paths, manual configs, or environment-specific tweaks anymore. That's why, during the migration, we audited the entire codebase for legacy shortcuts and rewrote them with cloud compatibility in mind.

- **What we did:** cleaned up hardcoded file paths, static URLs, and embedded configs. We introduced Spring profiles for clear environment separation, keeping dev, staging, and prod logic from bleeding into each other.
- **Example:** identified and rewrote over 100 FlexibleSearch queries that relied on outdated syntax or platform-specific quirks. This was critical to prevent query failures and data inconsistencies that could surface in the cloud environment, where validation is stricter and behavior is less forgiving.
- **The result:** a more consistent system that behaves the same way no matter where it runs, reducing surprises in production and making automated testing far more reliable.

Cloud compatibility reduces drift between environments, and that starts with stripping out the legacy logic your system quietly depends on.

DON'T WAIT TO MODULARIZE PRICING LOGIC

Once you're in the cloud, every tight coupling becomes harder to untangle, and the blast radius of small changes grows fast. In the client's case, the pricing logic was tightly coupled to the B2C Accelerator and difficult to adapt to telecom-specific workflows. That structure wasn't designed to handle multiple purchase models, and that limitation would've become a blocker in the cloud.

- **What we did:** migrated the entire pricing engine into the Telco Accelerator before cloud transition, moving it to a service-layer-based mechanism tailored for telecom. That shift decoupled pricing from the UI layer and enabled external, event-driven recalculation, which gave us more flexibility to serve different business models with cleaner orchestration.
- **The result:** a modular pricing service that could scale across use cases. And because it was cloud-aligned from the start, we avoided the pain of rearchitecting after the migration.

By modularizing pricing before the move, we gave the system room to evolve. With decoupled logic, it's easier to plug in new models, test changes in isolation, and extend features without touching the core checkout flow. That flexibility pays off long after go-live.

Key point: The “clean your house first” principle works, as real migration starts with the decisions you make about what comes with you.

What Makes SAP Commerce Cloud Migration Work

Getting your codebase cloud-ready is one thing. But making the migration work in practice — reliably, at scale, under real business pressure — is another. Once you're past pre-migration prep, success becomes a matter of how your teams operate, how your architecture holds up, and how well your release and monitoring setup absorbs the change.

Some actions set the foundation for long-term stability, others quietly introduce friction that snowballs later. Here are seven practical insights — tested across real migrations — that show what to do (and when) to keep the process on track.

CLEAN BEFORE YOU SCALE

Don't assume your cloud DB can absorb legacy data without impact.

Cloud infrastructure is more efficient, but also less forgiving when it comes to volume. What your on-premises database tolerated might throttle performance once you migrate.

Do archive what no longer serves your operational needs.

Before migrating a global luxury retailer to SAP Commerce Cloud, we archived years of historical order data that weren't tied to any active workflows. That one move boosted database performance by 40-50%, reduced replication delays, and made rollout velocity far more predictable.

Bringing everything into the cloud can slow you down and quietly rack up costs if you don't filter what really matters. Explore [where stale data introduces additional costs](#) and how to spot it early.

DESIGN FOR HYBRID COEXISTENCE

Don't underestimate the complexity of running legacy and cloud in parallel.

SAP Commerce Cloud migrations require legacy systems and new cloud services to run side by side for months. That takes planning at the integration level.

Do audit your integration landscape early.

In a project for a global retail and health & beauty group, we introduced a middleware layer to manage data flow between the legacy platform and SAP Commerce Cloud. This lets both systems run in parallel – syncing orders, inventory, and user data – without blocking the rollout.

Coexistence should be kept in mind from the start. If it's stable, the migration moves faster.

UPGRADE YOUR PIPELINES BEFORE THE PLATFORM

Don't bring brittle delivery workflows into the cloud.

The cloud won't fix deployment problems, but will surface them. Flaky tests, manual steps, or unclear ownership don't scale in a setup where small changes go live fast and often.

Do harden your CI/CD early and make it observable.

During a migration for a global financial analytics company, we reworked their pipeline architecture before touching the cloud: from modular builds to integrated quality gates. Tools like SonarQube and Veracode were added to enforce clean code and security checks at every step, shifting confidence left into the delivery cycle.

In the cloud, your pipeline is your safety net. The sooner your releases become boring, the faster your migration moves.

REVALIDATE YOUR THIRD-PARTY COMPATIBILITY

Don't carry legacy assumptions into cloud-based integrations

Legacy setups often mask edge-case behavior from third-party services – behavior that gets exposed in the cloud when retries, timeouts, and network conditions shift. Relying on old response patterns creates blind spots.

Do harden critical third-party flows before rollout.

We revisited the Adyen payment integration for a large retail group. Unexpected transaction responses were breaking flows that had worked on-premises. We refactored the logic to include fallback handling, improving stability and transparency during rollout.

Don't wait for cloud rollout to find out which API breaks first. [See what it takes to integrate like you mean it.](#)

MAKE PRE-PROD A MIRROR, NOT A SHADOW

Don't treat staging environments as rough drafts.

Even a minor drift in configs, service versions, or deployment logic between environments introduces false confidence. During cloud rollout, those gaps make failures harder to detect and debug under pressure.

I Do enforce environment parity across the board.

For a multinational beauty retailer migration, a failed go-live was traced back to mismatched configurations between staging and production. We solved it by fully aligning CI/CD pipelines and environment variables, ensuring consistent behavior across every deployment stage.

Matching environments don't eliminate risk, but they expose it early, when you still have room to fix it.

PLAN THE ROLLBACK BEFORE YOU NEED IT

Don't treat fallback as a post-failure reaction.

Cloud platforms give you more flexibility, but only if you plan for it. Too many teams focus on rollout velocity and forget that rollback paths must be defined, tested, and ready before day one.

I Do architect for reversible deployments from the start.

For a credit assessment company, we designed the rollout around parallel infrastructure, running both legacy and SAP Commerce Cloud clusters side by side on Amazon EKS. That setup gave the team a safety switch: if something went wrong, they could shift traffic instantly without disrupting business.

REDESIGN HEAVY OPS FOR CLOUD-GRADE PERFORMANCE

Don't rely on old usage patterns to predict performance in the cloud.

Legacy environments often hide inefficiencies behind overprovisioned hardware. In the cloud, where cost and speed scale together, bloated logic shows its cost fast, especially during bulk operations.

I Do optimize core processes with scale conditions in mind

For a global beauty and wellness retailer, we replaced full imports of 60,000+ promotions with a delta-import mechanism during migration. That switch cut sync time from 10 minutes to seconds and stabilized response times across high-traffic workflows.

You can nail the platform switch and still break your business logic if the data doesn't follow cleanly. Here's how to make data integrity part of the move, not a patch after.

After Go-Live: Fixing What the Migration Didn't Solve

You've migrated: everything is deployed, and the system runs. But you're not done yet. Post-migration, teams often find that some issues don't show up until real traffic hits. Workflows that have looked solid in lower environments can buckle under production pressure.

That's exactly what happened during a migration we led for a global luxury retailer. The move to SAP Commerce Cloud went smoothly. But in the weeks after go-live, performance dropped across key flows. Here are the patterns we uncovered — and what teams should keep on their radar once the "migration" is technically complete.

CRON JOBS DON'T OPTIMIZE THEMSELVES

Post-migration, legacy jobs can quietly become performance bottlenecks. Left unchecked, they consume resources, delay real-time tasks, and break the pace expected in cloud environments.

- **Client example:** delays in customer-facing operations, like search indexing, order syncs, and delivery updates, were traced back to legacy cron jobs running outdated queries on inefficient schedules.
- **How to fix:** re-index impacted tables, rewrite slow queries, and stagger execution times to avoid peak load collisions. Even standard jobs require rebalancing to match the cloud's runtime patterns.

CLOUD DOESN'T CATCH SILENT COSTS FOR YOU

After migration, APIs can still rack up usage costs, and third-party services may still introduce risk. Left unmonitored, both can quietly turn into critical problems.

- **Client example:** after migration, unoptimized Google Maps usage spiked operational costs, while delays in patching vulnerabilities in third-party components introduced potential exposure.
- **How to fix:** we switched to lower-cost service alternatives when usage spikes, and added temporary security mitigations until official updates land.

CLOUD MIGRATION DOESN'T AUTOMATICALLY SPEED UP YOUR PAGES

In the cloud, static resources don't get a free performance boost, but surface the inefficiencies that were always there. Content-heavy pages hit load-time ceilings fast, especially when they depend on heavy assets and render-blocking calls.

Client example: post-migration monitoring flagged sluggish load times on high-traffic product pages. Image-heavy layouts and oversized assets caused visible UX delays, even with the improved infrastructure.

How to fix: introduce lazy loading for secondary assets, compress media files, and monitor performance metrics per component.

IGNORING DB HYGIENE COSTS YOU AFTER THE MIGRATION

Even after a successful migration, bloated or poorly indexed databases can drag down performance. Search latency grows, back-office operations slow down, and basic workflows start to feel heavier than they should.

Client example: years of accumulated order data slowed down admin search, product updates, and reporting tasks — all caused by missing indexes and DB clutter that hadn't been addressed during migration.

How to fix: archive stale operational data, rebuild indexes on high-use tables, and monitor DB response times under load.

CLOUD MAKES CACHE BUGS LOUDER

On-premises, cache issues can hide behind long TTLs and server proximity. In the cloud, with distributed layers and external services, cache invalidation can become a potential blocker for consistency and speed.

Client example: after migrating to SAP Commerce Cloud, cache mismatches between the Spartacus front-end, Node.js middleware, and the SAP back-office led to outdated product data and inconsistent page states.

How to fix: implement coordinated cache reset flows across all layers. Add traceability to cache behavior, and treat invalidation as a first-class concern.

Even clean migrations can get bogged down by stale cache logic. [Learn how to rethink caching](#) when performance starts drifting.

Operational Habits That Shape Cloud Success

We've seen a pattern when teams think of cloud migration as a finish line. In reality, it's just the setup.

“ *What comes after — your day-to-day operations — is what defines long-term success.* **”**

From what we've seen, the most resilient platforms rely on operational habits that make cloud capabilities actually work.

HABITS OF HIGH-PERFORMING CLOUD TEAMS:

I PROACTIVE MONITORING

Logging and observability should be wired into every service from day one. Alerts, traces, and metrics are part of what gets shipped.

I CI/CD DISCIPLINE

Rollbacks, blue-green deploys, and strong test coverage are what keep velocity from turning into fragility.

I PERFORMANCE OWNERSHIP

When dev and ops both own performance, bottlenecks surface earlier and don't pile up in production.

I FEATURE FLAGS & CONTROLLED ROLLOUTS

These turn risk into something measurable. Testing in production becomes strategic when you can control exposure and minimize risk in real time.

I CROSS-FUNCTIONAL COORDINATION

In cloud setups, testing and product can't trail behind engineering. QA, devs, and business move in lockstep to avoid drift and misfires.

Strong Platforms Start with Strong Mindsets

By 2026, cloud readiness will be survival mode for SAP Commerce platforms.

“ *The platforms that thrive will be the ones treated not just as migrations, but as engineering and cultural upgrades.* **”**

That means starting with clean code and modular logic, optimizing what the migration didn't solve, and adopting habits that actually fit the cloud. Success will come if you can blend new infrastructure capabilities with how your teams think, ship, and recover under pressure.

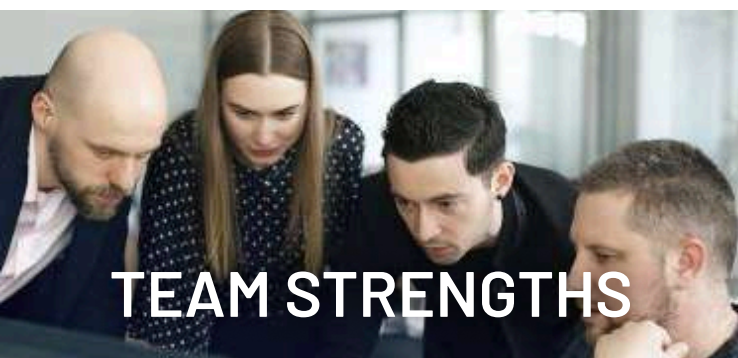
You can get far with the right mindset, solid engineering, and a clear migration strategy. But with a partner who's been through it, you can move faster, avoid the landmines, and build with more confidence at every step.

About Expert Soft

Expert Soft is a targeted ecommerce software delivery company, partnering with Fortune 500 companies and global corporations across the US and EU. With SAP Commerce Cloud and Java as our backbone, we know how to ensure scalable and high-performing solutions that can handle 1 mln requests per second, delivering a smooth customer experience.

Developing a payment engine that saved our client about \$100 million in operational expenses, ensuring multi-country platform support, adapting solutions for new market entry with tailored enhancements — these are just a few of the challenges our specialists tackle.

We aim to deliver more than a software system. We aim to deliver tailored solutions that maximize profitability within available resources. Our success is driven by:



TEAM STRENGTHS

- | All our engineers have a university background
- | Specialists excel their skills in our training LABs
- | Perfect English skills
- | Ready to help 24/7

CLIENTS

We work with corporations around the world with revenue of over \$20 billion and 150K+ employees.

APPROVALS BY AUDITS

Our ongoing work with corporations is consistently validated through rigorous audits, both by internal teams and Big 4 consulting firms.

HIGH-LEVEL SECURITY

Approved by assessments from global companies, who are leaders in their respective industries.

BUDGET EFFICIENCY

By carefully aligning technology investments with your business goals, we ensure optimal value and cost-effectiveness.

PROFESSIONAL TEAM

No offshore outsourcing and our team's average tenure of 4+ years means you get seasoned problem-solvers, not just coders.

EXPERT SOFT EXCELS IN

- | PAYMENT ENGINE
- | MICROSERVICES ARCHITECTURE
- | CONTENT MANAGEMENT
- | REDESIGN
- | E-COMMERCE PLATFORM
- | HEADLESS COMMERCE
- | MICRO FRONTENDS
- | MIGRATION&INTEGRATION

OUR TECH CORE



FRONT-END

HTML, CSS, JavaScript
(Angular, React, Vue,
Next, TypeScript,
Jquery), Spartacus



BACK-END

Java EE, Spring, SAP
Commerce (Cloud),
Node.JS.



DEVOPS

Docker, Kubernetes, CI/
CD



UX/UI DESIGN

UX Research, UI Design,
Figma, Adobe, Sketch



QUALITY ASSURANCE

Manual Testing, Test
Automation

TARGETED DOMAINS



SHARED PATHS, LASTING ECOM VICTORIES




LET'S TALK SOLUTIONS!




EKATERINA LAPCHANKA

Chief Operating Officer

kate.lapsenco@expert-soft.com

[+1 585 499 7879](tel:+15854997879) 

[+371 25 893 015](tel:+37125893015) 



PAVEL TSARYKAU

CEO & Founder
of Expert Soft

Let's connect 