JDD AGENCY

JDD Agency Guide to Legacy Migration

How to Decouple, Modernize, and Scale Before the Year Ends.

A JDD Agency Strategic Insight

The Cost of Doing Nothing

In 2026, legacy isn't just "old code" - it is an active barrier to AI integration.
If your data is locked in on-prem silos, you cannot leverage the LLMs and automation tools that your competitors are already using.

By 2026, technical debt will consume 60% of IT resources in companies that haven't modernized.

The Strategic Goal.

The Problem

Legacy systems operate as a "black box." A single change in one module (e.g., updating a pricing table) risks breaking the entire application (e.g., the checkout flow). This fear of breaking the system leads to "Deployment Paralysis," where updates happen once a quarter instead of once a day.

The Solution

We break the monolith into independent, self-contained building blocks (Microservices or Packaged Business Capabilities). These blocks communicate via APIs.

Why it wins: Replace the outdated block with a modern one.

The ROI

Scalability: Scale only the features getting heavy traffic (e.g., Black Friday search volume), not the whole server.

Resilience: If one service fails, the rest of the application stays online.

Speed: Deploy new features in hours, not months.

The JDD Migration Strategy

Our 6 R's Process.

Retire: *Turn off what isn't used.*
(Low Effort / High Savings)

Retain: *Keep what works and is secure.*
(Low Effort / Low Risk)

Rehost: *"Lift and Shift" to the cloud.*
(Medium Effort / Quick Wins)

Replatform: *Tinker to optimize for the cloud. (Medium Effort / Better Performance)*

Refactor: *Rewrite code for microservices. (High Effort / Maximum Agility)*

Repurchase: *Move to SaaS.*
(Variable Effort / Modern Standard)

We don't rewrite everything at once.
We recommend wrapping your old system in a new API layer, gradually replacing functionality piece by piece until the old system can be safely turned off.

The Toolkit

Operational Readiness Checklist.

- ☐ Have we mapped all dependencies?
- ☐ Is our data "clean" enough to move?
- ☐ Do we have a fallback plan if the migration stalls?
- ☐ Is the team trained on the new tech stack?

Pitfalls to watch.

Data Gravity:
Moving the app is easy; moving petabytes of data is hard. Plan for latency.

Scope Creep:
Fix the platform first, add new features second.

Stop patching. Start evolving.