// WHITE PAPER · PUBLISHED 16 SEPTEMBER 2026

Lift-and-Shift or Re-Architect: The Two That Modernized Later

What a rehost actually buys, what a refactor actually costs, and the line in the business case that decides both

Part 6 of 10 in *The Money Language Series*. Each piece stands on its own. The others: Part 1 · Part 2 · Part 3 · Part 4 · Part 5 · Part 7 · Part 8 · Part 9 · Part 10.

Jason Newell · MAX Research Collective · Report WPR-BIZ-007 · Published 16 September 2026
Licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0), https://creativecommons.org/licenses/by/4.0/
Original: https://www.maxresearchcollective.com/articles/lift-and-shift-or-re-architect.html
Cite as: Newell, J. (2026). Lift-and-Shift or Re-Architect: The Two That Modernized Later. MAX Research Collective, WPR-BIZ-007. https://www.maxresearchcollective.com/articles/lift-and-shift-or-re-architect.html

Every lift-and-shift I have worked on was justified the same way: move fast now, modernize later. Two of them modernized later.

Lift-and-shift earns its place, and the case for it simply has to say what it is buying. A rehost buys a change of location, and that is what the case should price. When the case instead claims savings that depend on a second project nobody has funded, it is fiction with a schedule attached.

The seven options

The industry framing is the seven Rs. Here they are with the part that decides the budget attached.

OPTION	WHAT IT MEANS	EFFORT	COST CHANGE	WHO IT SUITS
Retire	Turn it off	Lowest	Pure savings	Every portfolio holds more of these than anyone expects
Retain	Leave it where it is	None	None	Regulated, end of life, or economically fine where it sits
Relocate	Move the virtual machines wholesale	Low	Usually higher	A data centre exit under time pressure
Rehost	Lift and shift, architecture unchanged	Low	Often higher	Deadline-driven migration with decisions deferred
Replatform	Minor changes: a managed database, a container runtime	Medium	Modest improvement	The pragmatic middle
Repurchase	Replace it with software as a service	Medium	Shifts to subscription	Commodity functions such as HR, CRM and ticketing
Refactor	Re-architect for the target environment	High	Best long-run economics	Strategic systems with a long life ahead

Enterprise portfolios land at roughly 60 percent rehost, 20 percent replatform, 10 percent refactor, and 10 percent retire or retain; the mix moves with the estate. That distribution is fine. So what is the 60 percent actually buying? A location, mostly; the failure arrives when a program that is 60 percent rehost gets presented as a cost-savings program.

Newell, J. (2026). MAX Research Collective, WPR-BIZ-007. CC BY 4.0

Why rehosting usually costs more

Here is the arithmetic that gets skipped. On premises you sized for peak and for three years of growth, so the box ran at 15 percent average utilization and nobody minded, because the money was spent and the depreciation ran whether the capacity was used or not.

Rehost that workload and you rent the same oversized box, around the clock, at a price carrying the provider's margin and their option value on elasticity you are not using. The depreciation shelter goes too, which moves the spend above the EBITDA line; Part 1 is about why that single move changes how a decision reads.

Then come the costs that did not exist before, and they are the ones a pre-migration model never carries, because on the old estate they had no price.

NEW COST	WHY IT APPEARS
Data egress	Traffic that crossed a free LAN now crosses a metered boundary
Inter-AZ and inter-region transfer	Resilience patterns that were free inside one building are billed
Support plan	Often priced as a share of spend
Backup and snapshot storage	Previously absorbed by arrays you had already bought
Licensing uplift	Some vendors count cores differently on public cloud
Dual running	Both environments live through cutover, sometimes for months

The licensing line is the one that bites hardest. Commercial database and middleware licences frequently carry different terms on public cloud infrastructure — different core-counting rules, most often — so moving a workload without touching a line of code can multiply the licence bill. Check the licence terms before

the architecture review. Doing it afterwards has killed more than one business case at the eleventh hour, which is the most expensive hour available.

What re-architecting buys, and what it costs

Refactoring costs real money and real calendar time. Managed services, autoscaling, event-driven patterns and consumption pricing produce savings only when the application can genuinely scale down, and most applications written for fixed infrastructure cannot.

What you get is consumption pricing that tracks demand rather than peak, operational burden shifted to the provider, elasticity you can use instead of pay for, better resilience patterns most of the time, and a codebase your engineers want to work in. That last one is a retention argument, and it is worth making out loud rather than smuggling in.

What you pay is three to ten times the effort of a rehost depending on the system, a stretch of dual maintenance, a feature freeze or slowdown that almost nobody quantifies, and a fresh set of lock-ins to different services. The financial framing differs too, and the difference there is real rather than cosmetic.

	REHOST	REFACTOR
Nature of spend	Mostly operational project cost	A development project with defined scope
Book treatment	Usually expensed	Substantial portions may be capitalizable under ASC 350-40
Tax treatment	Deductible	Domestic development costs deductible now under Section 174A
EBITDA effect	Negative now, negative later	Depends on capitalization, and often frames better
Board framing	”We are moving”	”We are building an asset”

A refactor creates a durable asset. A rehost creates a change of address. That is why the same $4 million reaches a board in two different shapes, and why the technology leader who understands the difference presents it better. Part 3 covers the capitalization rules and the artifacts needed to support them.

The middle path

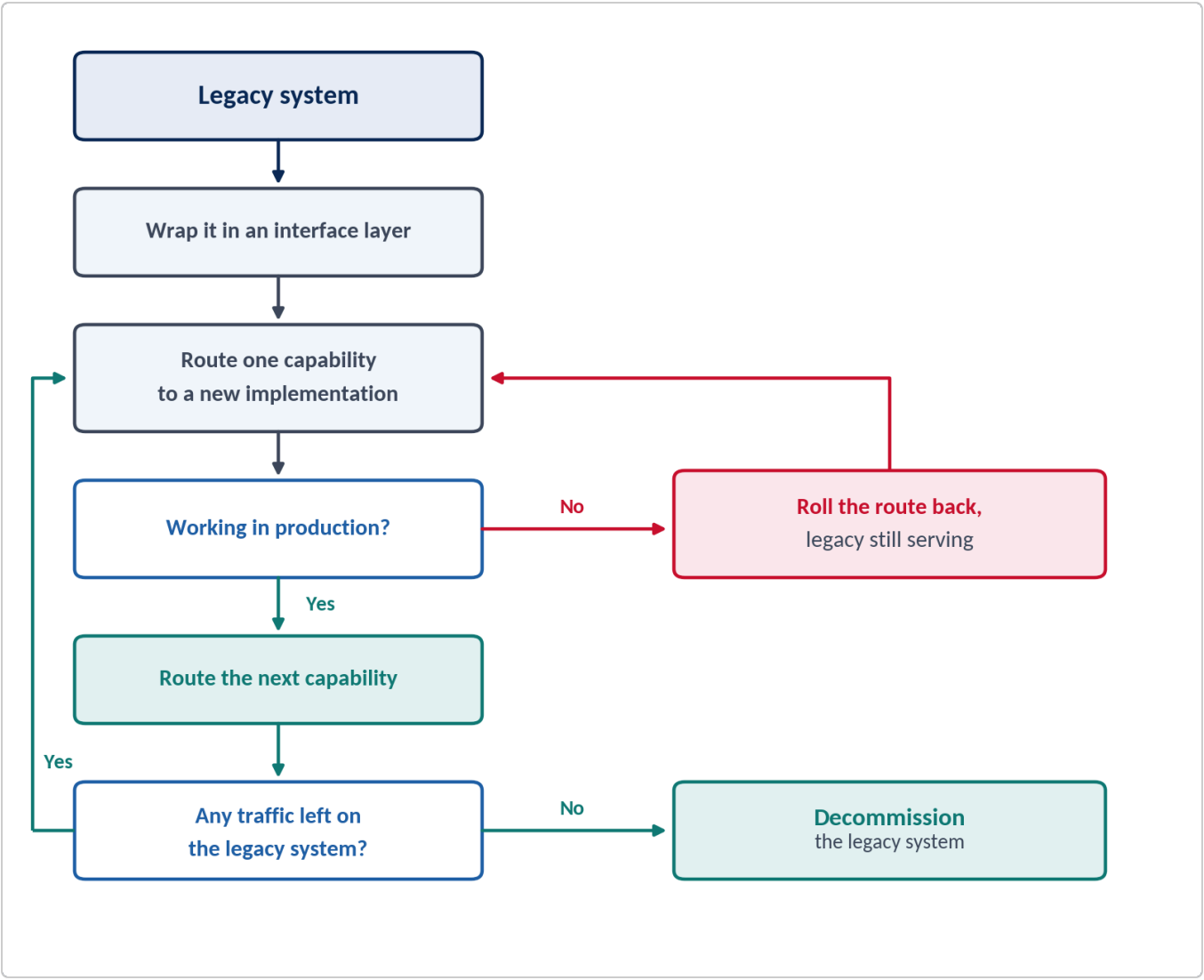


Figure 1. The strangler fig, drawn as what it is. Every arrow out of the failure gate goes back into the loop rather than out of it, which is the property that makes the pattern survive a bad increment.

The strangler-fig pattern replaces a system one capability at a time, with old and new running side by side behind a routing layer, until nothing routes to the old one.

It tends to beat both extremes for reasons that compound. Value lands continuously rather than at the end of a two-year program. You can stop at any point and still be ahead. Each increment is independently fundable, which matters a great deal under the authorization-and-probability test that replaced the project-stage model. Rollback is a routing change rather than a restoration. And it survives a change of leadership, which big-bang rewrites generally do not.

The cost comes in two parts: a longer stretch of running both systems, and the discipline to actually decommission the old one. I have watched the pattern fail once, and the failure was that nobody ever turned the old system off.

Sequencing with the data platform

The most common sequencing error is migrating applications first and the data platform second. Applications are easier to move and they produce visible wins, which is exactly why they go first.

The gravity is in the data platform. Leave it behind and every migrated application now reaches across a metered boundary for its data, so you have manufactured an egress bill and a latency problem together, and you will spend the next year explaining why the migration made things slower.

The order that works is plain enough: decide where the data lives, move it, then move the things that read it. It is less satisfying in the first quarter and cheaper in the fourth.

An honest business case

SECTION	WHAT GOES IN IT	THE TEST
Current state cost	Fully loaded: depreciation, licences, facilities, staff	Would finance recognize these numbers?
Target state run rate	Steady state after migration, with egress and support	Is the utilization assumption defensible?
One-time project cost	Engineering, dual running, tooling, partner fees	Is dual running in here?
Migration funding offset	Provider credits, counted realistically	Is this milestone-earned or assumed?
Accounting treatment	What is capitalized, what is expensed, the EBITDA effect per year	Has the CFO seen this?
Tax treatment	Section 174A deductibility, bonus depreciation on any hardware	Domestic or foreign development?
Break-even	The month cumulative cost crosses over	The month, rather than the five-year total
What we are not doing	The deferred modernization, named	Is it funded? If not, say so
Downside case	Demand halves, and the migration overruns by half	Does the case still hold?

The second-to-last row is the one worth insisting on, and it is the row most often deleted before the deck goes out. Who asked for its removal? That answer is usually more useful than the row. ”We will modernize later” belongs in the business case as an unfunded item, in writing. If it is really going to happen it has a budget line and a date (both, not either); if it has neither, the savings attributed to it come out of the model.

Newell, J. (2026). MAX Research Collective, WPR-BIZ-007. CC BY 4.0

A company I worked with rehosted 140 workloads in nine months to exit a data centre lease. The migration was well run and it hit its date. Cloud spend came in 22 percent above the on-premises run rate — disclosed at the start, and accepted — because the alternative was signing another five-year lease on a building they were leaving anyway. That is a good lift-and-shift case. It claimed a lease exit and it delivered a lease exit.

What I actually do

1. Run the retire pass before anything else. It is free money and it shrinks everything downstream, and somewhere between 5 and 15 percent of a typical estate is serving nobody at all (the range is that wide because almost nobody measures it until they have to).
2. Classify every workload against the seven Rs before proposing a program, then publish the mix. A program that is 70 percent rehost should say so on the second slide.
3. Check the commercial licence terms before the architecture review, every time, with no exceptions.
4. Move the data platform first unless there is a specific reason to do otherwise, and write the reason down when there is one.
5. Default to the strangler fig for anything strategic. Big-bang is for a hard external deadline and for nothing else.
6. Name the deferred work and its funding status inside the business case rather than beside it.
7. Model the downside before the board asks for it. A migration case that only works at plan is a forecast rather than a case.
8. Present the one-time cost and the run rate as two separate numbers, every time. Part 2 covers why that framing decides the outcome at some companies before anyone reads the total.

Glossary

TERM	DEFINITION
Break-even	The point where cumulative savings pass cumulative cost
Data gravity	The tendency of a large data set to make everything depending on it expensive to move
Dual running	Operating the old and new environments at the same time during a migration
Egress	What a cloud provider charges to move data out
Feature freeze	Pausing product work during a migration, a real cost and a rarely counted one
Refactor	Re-architecting the application for the target environment
Rehost, or lift and shift	Moving a workload without changing its architecture
Relocate	Moving virtualized infrastructure wholesale to an equivalent hosted platform
Replatform	Small changes to use managed services, without a redesign
Seven Rs	Retire, retain, relocate, rehost, replatform, repurchase, refactor
Strangler fig	Replacing a system one capability at a time behind a routing layer

References

- AWS, Migration Acceleration Program documentation and the seven Rs migration strategy framework
- Opsio, "AWS Migration Acceleration Program (MAP) Explained", for the portfolio distribution observations
- Martin Fowler, "Strangler Fig Application", the origin of the pattern

- FASB, ASU 2025-06 and ASC 350-40, on the capitalization treatment of refactoring work
- IRC Section 174A as amended by the One Big Beautiful Bill Act, 2025
- Flexera, 2026 State of the Cloud Report, on post-migration cost outcomes

Best next read

Part 7, Vendor Economics. A migration is where vendor terms get renegotiated and where multi-year commitments get signed. Part 7 covers licensing, support, commits, and the provider migration funding worth asking for before any of this is committed to.

The full series

PART	TITLE
1	CapEx, OpEx, and EBITDA: The Same Dollar, Two Different Stories
2	Reading the Financials: The Primer for Everyone Who Never Had To Care
3	Capitalize or Expense: Your Jira Board Is Now Audit Evidence
4	Depreciation and Useful Life: The Chip Outlives Its Economics
5	Placement Economics: The Right Answer, Applied Everywhere
6	Lift-and-Shift or Re-Architect: The Two That Modernized Later, which is this one
7	Vendor Economics
8	AI Spend: Three Budgets Pretending to Be One
9	AI and Headcount
10	Outsourcing After Section 174A

This series is an introduction for technology leaders. It is accounting and tax context rather than accounting or tax advice. Treatment is fact-specific. Model it with your CFO and auditor before you commit.

CAREER & LEADERSHIP WPR-BIZ-007

// ABOUT THE AUTHOR

Jason Newell leads organizations and people in technology, and is a researcher, deep technologist and builder — from a CRAY to agentic AI. He founded MAX Research Collective to build AI for regulated industries that runs where its owner has control, with healthcare as its strongest focus. About Jason → Connect on LinkedIn →

Newell, J. (2026). MAX Research Collective, WPR-BIZ-007. CC BY 4.0