



BEYOND LIFT AND SHIFT

The Operating Model Never Made the Journey

Why cloud migrations move every workload except the one thing that determines whether they pay off — and the structural forces that keep it grounded

Giovanni Leonardi

August 2017

Contents

1. Executive Summary	3
2. The morning the migration was "done"	3
3. The honest case for moving first and changing later	4
4. Tracing the roots	5
4.1 The business case was a data-centre exit, not a transformation	5
4.2 Ownership sat with infrastructure, and Conway's Law did the rest	5
4.3 The accounting model fought the consumption model	5
4.4 Governance and procurement were built for a world of fixed assets	5
4.5 The operating model is invisible, and invisible things get deferred	6
5. A composite worth sitting with	6
6. What a cloud-ready operating model actually asks for	7
7. The objection worth taking seriously	8
8. What the pattern tells us about transformation	8

The Operating Model Never Made the Journey

Transformation · No. 211

Originally written August 2017 | Re-edited and re-rendered for the WhereBeyond Edition, 2025

"The building can be handed back. The operating model comes with you, or nothing really does."

1. Executive Summary

Across the last few years, "move to the cloud" has become one of those objectives that no board argues with and few examine closely. The migrations are real: workloads leave the data centre, the lease is not renewed, the press release is written. And yet a pattern recurs with unsettling regularity. Twelve or eighteen months after the last server is decommissioned, the organisation finds that its cloud bill is higher than the business case promised, its releases are no fewer weeks apart than before, and its people are doing more or less what they always did — only now they do it against a rented estate they understand less well than the one they owned.

The workloads migrated. The operating model did not. This essay is an attempt to trace why that gap is so common, so durable, and so rarely named for what it is. The short version is that lift-and-shift is not a failure of ambition or of engineering; it is the entirely rational product of the way these programmes are funded, owned, governed, and measured. The operating model — how work is funded, who runs what, who decides, and against which guardrails — is the least visible part of the estate and therefore the easiest to leave behind. Until that part makes the journey, an organisation has not migrated to the cloud. It has merely relocated its data centre and kept the receipt.

2. The morning the migration was "done"

There is a particular kind of all-hands meeting that marks the end of a large migration. The final tranche of workloads has moved. A slide shows a map of the old estate with a satisfying line drawn through it. Someone from infrastructure is thanked, deservedly, for eighteen months of unglamorous, weekend-consuming work. The lease on the second data centre will lapse at the end of the quarter. By any reasonable definition of the programme that was chartered, the thing is finished.

Then the quarter turns, and the questions start. Why is the monthly cloud invoice running a third above the model? Why does a change to the pricing engine still take six weeks to reach production, when the whole point of this was speed? Why are there now three people whose full-time job is to reconcile a bill that used to arrive once a

year from a hosting provider and get paid without comment? None of these questions has a satisfying answer, because none of them was in scope. The programme was chartered to move workloads and vacate a building. It did exactly that. What it was not chartered to change was the way the organisation worked — and that, it turns out, was the part that determined whether any of the promised benefits would actually arrive.

We should be precise about what "lift and shift" moves and what it leaves. It moves the workload: the application, its data, its dependencies, re-hosted onto rented infrastructure. It leaves almost everything else. It leaves the funding model, in which infrastructure is a capital project approved once and depreciated over years, now colliding with a consumption meter that ticks every hour. It leaves the team topology, in which one group writes software and throws it over a wall to another group that runs it. It leaves the governance, in which a change advisory board meets weekly to approve deployments that the platform could now gate automatically. It leaves the decision rights, the skills, the incentives, and the mental model of a server as a pet with a name rather than a unit of capacity to be created and destroyed at will. The tin changed hands. The way of working stayed exactly where it was.

3. The honest case for moving first and changing later

It would be too easy, and unfair, to treat lift-and-shift as simple negligence. The practitioners who run these programmes are not fools, and the strongest version of their reasoning deserves to be met head-on rather than caricatured.

The case runs like this. Data centre leases and hardware refresh cycles do not wait for an operating-model debate. When a lease expiry or an end-of-support deadline is bearing down, the responsible thing is to get the estate onto a modern platform first and optimise afterwards. Re-architecting four hundred applications before you move any of them is a recipe for moving none of them; the perfect migration is the enemy of the migration that actually happens. Moving first also buys optionality — once a workload is in the cloud, you can modernise it incrementally, at a pace the business can absorb, rather than betting the programme on a big-bang redesign. And there is a genuine skills argument: teams learn the platform by living on it, so getting them there, even crudely, starts the clock on the learning that real modernisation depends on.

Every clause of that argument is true. The mistake is not in the sequencing — move first, change later can be entirely sound. The mistake is in treating the second half as optional, or as something that will happen on its own once the workloads have landed. It will not. The forces that produced lift-and-shift do not evaporate at the moment of migration; they persist, and they quietly convert "change later" into "change never." To see why, we have to look past the technology and at the structures that surround it.

Lift-and-shift is rarely a technical decision that went wrong. It is an operating-model decision that was never consciously made — the default that remains when no one is chartered, funded, or measured to change how the work is done.

4. Tracing the roots

If the pattern were merely a matter of teams not trying hard enough, it would not recur so faithfully across organisations that share almost nothing else. It recurs because several structural forces push in the same direction, and each of them is individually reasonable.

4.1 The business case was a data-centre exit, not a transformation

Follow the money back to the origin of most migration programmes and you rarely find a benefits case built on agility, resilience, or product velocity. You find a real-estate and hardware case: a lease to exit, a refresh to avoid, a run-cost line to reduce. That framing is what got the programme funded, and a business case is not a neutral document — it is a set of instructions about what success means. A programme justified by exiting a building is measured on whether the building is exited. Nobody asked it to change how software is delivered, so nobody funded, staffed, or governed that change. The operating model was out of scope from the first spreadsheet, and scope, once set, has enormous inertia.

4.2 Ownership sat with infrastructure, and Conway's Law did the rest

Migrations are almost always owned by the part of the organisation that owns servers. This is sensible — they know the estate — and it is also decisive, because the shape of what a team produces tends to mirror the shape of the team that produces it. An infrastructure-owned programme produces an infrastructure outcome: the machines move, competently, and the questions that live outside infrastructure's remit — how teams are organised, how funding flows, how decisions are made — are not so much refused as never raised. Conway observed half a century ago that organisations design systems that copy their own communication structures. A migration run by one silo will reproduce the silos, faithfully, on rented hardware.

4.3 The accounting model fought the consumption model

For decades, infrastructure was a capital asset: buy it, capitalise it, depreciate it, and treat the annual run cost as a fixed and largely invisible overhead. Cloud replaces that with a variable operating expense that moves with usage, hour by hour. The two models are not merely different line items; they imply different behaviours. A capitalised world rewards buying headroom — over-provision once, sleep easy for three years. A consumption world punishes exactly that habit, because the headroom now bills continuously. Organisations that migrated without changing how they budget, forecast, and hold teams accountable for consumption simply ran their old capital-era instincts against a meter. The result is the over-sized, always-on estate that produces the surprise invoice: instances specified like the physical boxes they replaced, running through the night to serve no one.

4.4 Governance and procurement were built for a world of fixed assets

The change advisory board, the procurement gate, the architecture review — these were engineered for a setting where change was slow, expensive, and hard to reverse,

and where the appropriate response to risk was to inspect every change before it happened. Cloud inverts the economics of change: it becomes cheap, fast, and reversible. But the governance did not invert with it. So the organisation ends up with a platform capable of deploying a hundred times a day, throttled by a weekly meeting designed to approve a deployment a month. The controls that made sense against fixed assets become the binding constraint on the very agility the migration was supposed to unlock.

4.5 The operating model is invisible, and invisible things get deferred

Underneath all of this sits the simplest force of all. You can point at a rack of servers; you can watch them power down. The operating model has no such physical presence. It is the sum of a hundred tacit agreements about who does what, who pays, who decides, and what "done" means. Because it cannot be seen, it cannot easily be project-managed, and because it cannot be project-managed, it loses every scheduling contest to work that can. Faced with a visible deadline (the lease) and an invisible ambition (the operating model), a programme will always, and rationally, spend its last month on the deadline. "Change later" is not a lie anyone tells; it is what is left on the cutting-room floor when the visible work consumes the time.

5. A composite worth sitting with

Consider a mid-sized general insurer — the details invented, the shape drawn from life. It commits in early 2016 to vacate a leased data centre by the end of 2017, a hard date driven by the lease. Roughly four hundred workloads are in scope. The programme is owned by infrastructure, funded on a run-cost-reduction case that projects a twenty per cent saving against the current hosting spend, and governed by the existing weekly change board.

The migration itself is a success on its own terms. By the summer of 2017, some three hundred and sixty workloads have moved; the building will be handed back on schedule. Then the operating reality arrives. The cloud run rate is landing around thirty-five per cent above the modelled figure, not because the platform is expensive but because the workloads were re-hosted at the size and duty cycle of the physical servers they replaced — provisioned for a peak that occurs twice a year, running twenty-four hours a day, in an environment that would happily have scaled them down every night if anyone had been asked to make it do so. Change lead time, meanwhile, is unmoved at around six weeks, because the same change board reviews the same deployments under the same rules; the platform can deploy in minutes, but the organisation cannot decide in fewer than six weeks. And a new cost has appeared that the case never contemplated: a small standing team assembled to interpret, allocate, and argue about a bill that arrives monthly and varies, where the old world sent one predictable invoice a year.

Dimension	What lift-and-shift moved	What it left behind
Workloads	Re-hosted onto the platform	Sized and scheduled like the old tin
Funding	A one-off capital case, spent	A consumption meter no one owns
Delivery	Faster technical deployment	A six-week decision cycle, intact
Run	New place to run the software	The same "build it, throw it over the wall" split

Nothing in this story is a technical failure. Every individual decision was defensible. The programme did what it was chartered, funded, and measured to do. The disappointment is entirely a function of the gap between what moved and what stayed — and that gap was designed in, silently, at the moment the business case was framed as a building to exit rather than a way of working to change.

6. What a cloud-ready operating model actually asks for

If the operating model is the migration that matters, it is worth being concrete about what changing it involves, because vagueness here is part of how the work gets deferred. A cloud-ready operating model is not a slogan; it is a set of specific shifts, most of which have nothing to do with technology.

1. Fund products, not projects. A consumption platform cannot be run on a one-off capital grant. Someone has to own the run rate as a living number, forecast it, and be accountable for it — which means moving from project funding, approved once, to persistent product funding that carries responsibility for cost as well as feature.
2. Make teams run what they build. The wall between those who write software and those who operate it is what keeps run cost and reliability invisible to the people best placed to change them. Giving delivery teams responsibility for their own workloads in production is what turns "the bill is someone else's problem" into a design consideration.
3. Replace gates with guardrails. The point of cloud is that change is cheap and reversible; governance should exploit that, not fight it. That means encoding policy into the platform — automated checks, standards enforced in the pipeline — so that the common case flows without a meeting, and human review is reserved for the genuinely novel.
4. Move decision rights to where the information is. A six-week lead time is usually a decision-rights problem wearing a process costume. If a team can see the consequence of a change and can reverse it, it can be trusted to make it. Centralised approval made sense when change was slow and irreversible; it is pure drag when change is neither.
5. Budget for capability, not just capacity. The skills that run a cloud estate well — sizing to demand, automating the routine, reading a consumption bill as an engineering signal — are not the skills that ran the old data centre. If the

programme funds the move but not the learning, the estate will be run by yesterday's instincts against today's meter.

None of these is exotic, and none is free. Every one of them touches funding, org design, governance, or incentives — precisely the territory an infrastructure-owned, deadline-driven migration is neither chartered nor equipped to enter. Which is exactly why they get left for later, and later rarely comes.

7. The objection worth taking seriously

The strongest counter to all of this is not that the operating model does not matter, but that it can safely follow. Get the workloads across, the argument goes, and the pain of the surprise bill and the six-week lead time will itself create the mandate to change the operating model — reality will force the reckoning that a slide deck never could.

There is something to this. Pain does concentrate minds, and it is true that some organisations only take the operating model seriously once the invoice makes it impossible to ignore. But the sequencing carries a trap that is easy to miss. The window in which change is cheapest is during the migration, when budgets are open, attention is high, teams are already being disturbed, and the platform is still being shaped. Once the programme declares victory and disbands, that window closes. The funding is spent, the specialists have rolled off, the temporary licence to disrupt has expired, and the operating-model change now has to be relaunched cold, as a fresh initiative competing against everything else — with the added difficulty that the estate is now a rented replica of the old one, its bad habits poured into infrastructure-as-code and thereby made more durable, not less. "Change later" does not merely defer the work; it makes the work harder, because you have spent the migration hardening the very structures you now need to unpick.

"An organisation that migrates its workloads but not its operating model has not moved to the cloud. It has rebuilt its data centre somewhere more expensive and called it progress."

8. What the pattern tells us about transformation

The reason this essay is worth writing is that cloud migration is only the most legible instance of a far more general failure, and seeing it clearly here helps us recognise it elsewhere. The recurring shape of unsuccessful transformation is this: the organisation changes the visible, tractable, deadline-bearing thing — the platform, the tool, the system — and leaves unchanged the invisible, intractable, deadline-free thing — the operating model, the incentives, the decision rights — and then expresses surprise that

the benefits, which lived entirely in the second thing, did not arrive.

We have seen this pattern before under other names. It was there when organisations bought enterprise systems and configured them to replicate the processes they were meant to replace. It was there when shared-service centres centralised transactions without changing the accountabilities around them. It is there, now, in cloud. The technology is always the part we can point at, procure, and project-manage, and so it is always the part that gets done. The operating model is always the part that determines whether the technology delivers anything, and so it is always the part left for a phase two that the programme's own structure ensures will never quite begin.

If there is a single lesson worth carrying out of the cloud experience, it is that the operating model is not the soft, downstream, change-management afterthought it is usually treated as. It is the actual object of the transformation. The workloads were never the point; they were the visible proxy for a change in how the organisation works, and when we mistake the proxy for the goal, we get exactly what we are now seeing across the industry — estates that migrated flawlessly and organisations that did not move at all.

The building can be handed back. The operating model comes with you, or nothing really does.