CLOUDBOLT

EBOOK · 18 MIN READ

The lesson of the wasted cluster.

Why the hardware crunch isn't your real problem, and why forced automation won't fix it.

A six-perspective look at the moment the dashboard says the cluster is full, Finance is bracing for a multimillion-dollar ask, and **the actual hardware is running at 18% CPU**. The shortage isn't silicon. It's trust.

CLOUDBOLT · 2026

INSIDE THIS EBOOK

Six rooms, one cluster, and the trust gap nobody owns.

A platform team has a $4M hardware ask ready. The dashboard tells a different story. Across six perspectives, here is what is really happening inside the war room.

THE SCENE OF THE HARDWARE CRIME

The cluster is "full."

That's what the dashboard says. And that is what everyone has agreed to as their shared reality.

The platform team is preparing a multimillion-dollar hardware ask. Finance is bracing for another unplanned exception. Procurement is warning that the quote may not survive the week. And app teams are insisting their buffers are justified.

The macro story on the state of hardware in the face of rapid AI expansion is ugly enough to make the ask feel inevitable. AI infrastructure demand has pushed hyperscaler spending into surreal territory. Reuters reported that major technology firms are turning to debt markets to fund surging AI infrastructure, with industry-wide AI spending projected above **$700 billion in 2026**. Meta is also reportedly arranging roughly **$13 billion** in financing for an El Paso data center, part of a broader Big Tech data center buildout tied to AI demand.

Memory and storage are not behaving like background costs anymore; they've become front of the forehead migraines. Gartner projects a **130% surge in combined DRAM and SSD prices** by the end of 2026. The Register reported that DRAM and NAND flash prices were surged in Q1 2026, with NAND flash projected up 55–60% versus Q4. Avnet notes that AI data centers are consuming memory at a rate that is pushing DRAM and NAND prices radically higher, with enterprise SSDs leading storage demand growth.

So when a platform leader walks into Finance and says, "We need more hardware," the story sounds credible and unavoidable.

But the story is often incomplete because in too many enterprises, the cluster is not full of usage.

It's full of requests.

CPU usage is sitting in the teens. Memory is below 40%. The cluster shows 90% utilization or more. The scheduler has nowhere to place new work, but the actual hardware is nowhere near being exhausted.

And that's the crime scene where this eBook really begins, and across the next few chapters, we will look at it from every angle.

SIX ROOMS, ONE CLUSTER

PLATFORM TEAM
sees a capacity crisis.

APP TEAM
sees reliability insurance.

FINANCE
sees a capital request.

PROCUREMENT
sees a supply chain trap.

CFO
sees a symptom nobody has fixed.

DASHBOARD
sees the truth.

SPOILER

The real shortage is not hardware or technology at all.

> **The cluster is not full of usage.**
> **It's full of requests.**

THE OPENING LINE OF EVERY HONEST CAPACITY REVIEW

PERSPECTIVE 01

01

The platform team's problem.

A capacity spreadsheet. A vendor quote. A directional price that may be stale before the meeting ends. And the number at the bottom of the expansion plan that nobody wants to read out loud.

The war room is too quiet. That's how you know the number is bad.

A capacity spreadsheet is open. So is the dashboard and a vendor quote. Someone from procurement has already said the price is "directional," which is a polite way of saying it may be stale before the meeting ends.

The cluster is almost out of room. Not out of CPU or memory. **Out of room.**

The distinction matters, but not yet. Not while everyone is staring at the number at the bottom of the expansion plan.

> $4 million - that's the ask.

The platform team has the justification ready. Hardware lead times are ugly, memory costs are rapidly rising, and storage inventory is tightening. AI demand has swallowed the supply chain. Finance isn't going to like it, but they will almost certainly understand it.

And that's the seductive part: **the hardware story is true enough to hide the deeper problem**…the one that's on the other open screen that quietly screams the rest of the story:

CPU UTILIZATION

18%

Workloads idle through most of the day.

MEMORY UTILIZATION

38%

Headroom on every node, hour after hour.

CLUSTER ALLOCATION

94%

The scheduler has nowhere to place new work.

VERDICT

The cluster looks full because requests have consumed the schedulable space. The workloads themselves are not using anything close to what they reserved.

Everyone in the room knows the implications of that revelation, but many want to not even acknowledge it. The team was locked and loaded to engage with Finance on fixing a hardware problem. However, if this is a requests problem, the next meeting is with app teams instead. And nobody wants that meeting because app teams do not give back capacity simply because a dashboard says they should.

The apps teams offer a barrage of push back. They ask what happens if the recommendations and subsequent automation is wrong. They ask who owns the pager. They ask whether rollback is instant. They ask why the platform team gets the savings while they inherit the risk.

None of those are bad questions. They're enterprise questions by nature.

This is when a naive vendor usually enters the room and confidently says, "Just turn on autonomous rightsizing. Let the system act. Stop negotiating with every team. Trust the algorithm to rapidly and retroactively course correct." What could happen?

That may work in a lab, a startup, or a small environment with tight ownership and high cultural trust (or where an occasional app failure is tolerated), but it is completely antithetical to how most large enterprises start operate – at least until a necessary level of trust has been built.

The platform team knows this. They are not afraid of automation. They are afraid of destroying the trust automation needs to survive and thrive.

Forced automation says: we demand full control now.

Adaptive autonomy says: we will earn the right to take more action continuously over time at the pace you define.

And that's a profound difference because it changes the question on the table.

The platform team did not walk into the room to debate philosophy. They came to prepare a hardware ask. But now the dashboard has exposed something harder.

Should they really even be asking Finance for money?

Or should they first be asking the tougher question about why the company cannot actually maximize and utilize the capacity it already owns?

PERSPECTIVE 02

02

The app team’s logic.

Two years earlier, nobody called it waste. They called it the thing that finally let them sleep through the night.

Two years earlier, nobody called it waste.

At 2:07 a.m., there was an outage. The app lead woke up to a page, joined the incident bridge, watched pods cycle, saw memory pressure, and listened while Slack filled with questions from people who were not going to help fix it and despite best intentions only made the matter worse.

The team struggled to stabilize the service and eventually righted the ship and then vowed to never let that happen again.

Coming out of the crisis, they did what any sane team would do: They raised the memory request, and then they added buffer. Nobody objected. Why would they?

> The cost of being under-provisioned was immediate and personal. The cost of being over-provisioned was invisible and paid by someone else. That's the perverse incentive structure.

Over time, the namespace became a storage unit filled with a collection of old fears. One OOMKill here. One peak event there. One "temporary" request increase that became permanent. Another to accommodate the monstrous bloat of a Java app on start-up. One copied config from a heavier service. And a little extra safety margin for good measure because nobody gets promoted for shaving memory too close. Everyone on the app team slept well at night knowing that they had effectively built a system designed to ensure that the pager alert never came anymore.

But then the platform team arrives with a recommendation that receives all the welcome of a Baby Ruth bar floating in a public swimming pool.

"We see that your workloads are over-requested. The data is clear that you are nowhere near using what you have reserved. We can reclaim capacity."

To the platform team, this sounded completely rational and they expected to be congratulated for their brilliant insights. To the app team, it sounds like someone wants to **remove the airbags because the car had not crashed recently**.

But then the platform team goes one step farther and says, "We have just the solution to solve this problem and optimize capacity going forward." And they introduce a vendor that says their platform forces fully autonomous rightsizing from day one. The app team need not worry. The model is smart, the data is strong, and the savings are obvious. Just trust them.

That is precisely the moment when the app team stops listening. Not because they hate automation, but because the proposal skips the part where trust is earned.

The app team owns the SLO, the customer impact, and the incident review and response. If automation is wrong, the vendor doesn't get paged. The platform savings do not explain the outage to the VP. The algorithm does not sit in the postmortem and take the heat if there's an incident. It's the app team with team members' names in bright neon lights for everyone to scorn and ridicule.

So when the app lead asks, "What happens if this breaks?" (and it invariably will), that is not resistance. In fact, it's the right question.

No one likes to contort their operations to do something in an unnatural way simply because that's how a vendor's system makes them do it. And that's especially true when it comes to automatically optimizing a Kubernetes environment. That's Forced Autonomy and most enterprises will have a visceral reaction to it.

Adaptive Automation respects the enterprise process. It doesn't start by demanding full control. It begins by showing the recommendation, the usage history, the safety margin, the rollback path, and the ownership model. It gives teams a way to see the system behave before asking them to trust it.

That is how trust forms and builds, not through a mandate, but through repeated proof and success. The app team is not irrational. They are responding perfectly to the system they are in. The rational move has always been to over-request.

Until the system changes, why would they stop?

PERSPECTIVE 03

03

The Fortune 500 ultimatum.

"Adopt rightsizing or fund your own hardware." It was short, which made it land harder.

The email was short, which made it land harder.

"Adopt rightsizing or fund your own hardware."

At first, the app teams treated it like it was just blustery theater. Perhaps it's simply a pressure tactic or a budget scare. The kind of message that sounds dramatic on Monday and gets softened by Friday after three leadership reviews and a meeting called "alignment."

Then, as no one walked it back, the Slack channels lit up.

Is this real? Can they do that? What does "fund your own hardware" mean?

And then a litany of "what about"s were rapid-fired: What about customer-facing apps? What about regulated workloads? What about exceptions?

Every team thought its situation was the exception, which only further highlighted the problem. For years, rightsizing had been treated as a good idea with no real consequence. Everyone supported it in principle while delaying it in practice as they insisted on another review, more data, another delay, a different exclusion. And because hardware could always be bought at reasonable price levels, the organization tolerated the delay.

But this time, the platform leader changed the rules. The message was not: "We are turning on full autonomous rightsizing tomorrow." That would have been a mistake that led to mass revolt. Instead, the message was sharper and smarter:

"The era of invisible over-requesting is over. If a team wants to keep a large buffer, fine. Show the evidence. Own the cost. Explain the risk. If a team wants an exception, fine. Make it visible. If a business unit wants dedicated capacity, fine. Fund it."

That completely changed the conversation. The platform leader was not punishing app teams. He was simply ending the **subsidy**. For years, shared infrastructure had let everyone pretend capacity was free. The platform team carried the bill. Finance stomached the surprise. Procurement held their noses. App teams carried the reliability risk.

Now the ultimatum connected those pieces, but it also avoided the trap of forced automation. The goal was not to make every workload autonomous overnight. The goal was to make rightsizing **unavoidable** and automation **adaptable**. Teams would march forward building trust at their speed on their path toward increased and eventually full autonomy.

In an enterprise, automation maturity is not a slogan.

It is a negotiation with trust. The Slack storm did not disappear, but the tone changed. That's when everyone realized the old rules were gone and the new reality settled in. He's serious.

PERSPECTIVE 04

04

The cluster that was full.

No catastrophic alert. No wall of red. Just three numbers refusing to tell the same story.

The dashboard was almost boring.

That was the disturbing part.

No catastrophic alert. No wall of red. No obvious failure state. Just three numbers that refused to tell the same story. Instead, it hid the truth.

CPU: 18%. Memory: 38%. Requests: 94%.

Conclusion: The cluster was full.

Except it wasn't.

The scheduler was constrained, but in fact, the hardware was not. But based on these optics, the enterprise was preparing to buy additional capacity at previously unimaginable premiums. Capacity it already had but could not safely use.

"Safely" is the whole story.

This is why the "just automate it fully and immediately" crowd misses the point.

Unused capacity is not automatically reclaimable capacity.

In a real enterprise, reclaimable capacity requires trust. Trust that recommendations are conservative and workload behavior is understood. Trust that automation will not flatten every application into the same policy. Trust that rollback works and that app teams will not be blamed for participating.

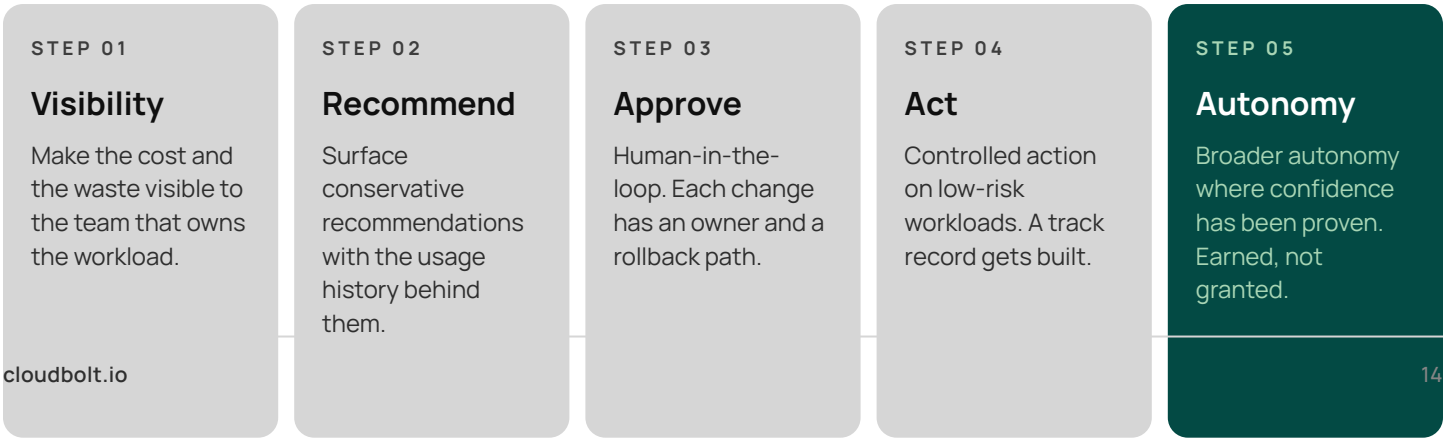
Without trust, the dashboard is an accusation. But when trust is built, it becomes a map.

Every over-requested workload is not just waste. It is a different trust problem.

- One team needs visibility.
- One team needs proof under load.
- One team needs human-in-the-loop approval.
- One team needs exception handling.
- One team is ready for automation now.
- One team might be ready in ninety days.

Forced automation treats them all the same. Adaptive automation does not.

That is the unlock.



The goal is not to avoid or delay automation and full autonomy. The goal is to earn it in the right order. Start with recommendations and let teams see the evidence. Preserve rational buffers while automating low-risk workloads first. Prove rollback. Build a track record. And then expand as confidence grows.

Trust compounds when automation behaves well, but it collapses when automation outruns the organization.

The platform engineer looks back at the dashboard and notes that the numbers have not changed. But the meaning has.

They are not out of capacity. They're out of trust.

THE DASHBOARD'S NUMBERS HAVEN'T CHANGED. THE MEANING HAS.

PERSPECTIVE 05

05

The two doors.

Back in the war room, the choice is finally clean. There are two doors. Everyone wants a third but there is none.

Back in the war room, the choice is finally clean. There are two doors. Everyone wants a third but there is none.

Door one is Finance and it is beguiling with its familiar process, proposal, and approval chain. The handle to that door is well worn through repeated use.

The platform team can simply walk in with the hardware quote, the lead times, the market data, and the growth forecast. Sure, the ask will hurt, but Finance will understand it and they won't resist it.

Door one is dangerous because it converts an operating model failure into a capital request. Buy more hardware, add more capacity, give the scheduler breathing room, absorb the price increase, and then wait out the lead time and move on.

For a while, it works.

Then the new capacity lands. Requests expand. Buffers grow. The cluster fills again. Next year's baseline is bigger, and nothing about the behavior has changed.

Door one buys time by kicking the same old capacity can down the rode.

It also buys the next version of the same (if not worse) problem.

DOOR ONE · FINANCE

Buy more hardware.

Walk in with the quote, the lead times, the market data, the growth forecast. The ask will hurt, but the process is well worn. Finance will understand it.

- Converts an operating-model failure into a capital request
- For a while, it works. Then the new capacity lands.
- Requests expand. Buffers grow. The cluster fills again.
- Next year's baseline is bigger. Behavior hasn't changed.

DOOR TWO · THE APP TEAMS

Stop wasting what you already own.

Slower. Messier. More political. No purchase order fixes it. Start where each team is. Make cost visible, recommendations explainable, rollbacks real, ownership explicit.

- Make the cost of buffers visible to the team that owns them
- Conservative recommendations with usage history attached
- Rollback is real, instant, and rehearsed
- Automation expands where the system has earned the right

Door two, on the other hand, is the app team's.

It's slower, messier, more political. There is no purchase order that fixes it.

And there is only one way to walk through it. Start where each team is. Make the cost visible, recommendations explainable, rollbacks real, ownership explicit, and exceptions possible in specific instances. Then, increase automation and eventually move to full autonomy when and where the system has earned it.

> **Door one** says: the world changed, so we need more money.
>
> **Door two** says: the world changed, so we need to stop wasting what we already own.

Adaptive automation is what makes door two walkable. Because the answer is not manual forever and it's not full autonomy tomorrow. It's automation that moves at the speed of trust.

Which door would you want to walk through?

PERSPECTIVE 06

06

The enterprise that got it right.

Back when the initiative was introduced, nobody wanted to participate. That part always gets edited out of the success story.

Back when the initiative was introduced, nobody wanted to participate.

That part always gets edited out of the success story. But a few months down the road, everyone remembers a more positive version. The company reclaimed capacity. The app teams cooperated. The hardware ask got smaller. The rightsizing program worked.

But originally, it was stuck in a seemingly endless loop.

The data and recommendations were there, and the waste was obvious. So things slowed down just enough to avoid a backlash.

Then things started moving again once teams accepted recommendations. They saw that nothing broke. A rollback path was tested while a conservative resize held under load. A skeptical app owner saw firsthand that the system preserved buffer instead of blindly cutting everything to the bone.

Eventually, as trust grew and began to accelerate, the teams that had stalled the effort were leaning in.

The company reclaimed roughly **30% of targeted capacity**, and now they were targeting an even higher level of efficiency for Phase Two.

No drama. But more importantly, no outage story poisoned the well. There were no cautionary tales. Instead, the automation implemented earned the right to do more.

As expected, the long-tail hardware crunch didn't disappear. Costs were still rising and lead times were still painful. Growth still needed funding, but the Finance conversation dramatically changed. The platform leader was no longer asking for money because the old model had failed. Instead, he was asking for more capacity to grow after proving the company had attacked the waste first.

And that's a very different conversation.

> Finance does not hate spending money. Finance hates funding symptoms instead of cures.

This time, the enterprise was not just buying capacity.

It was fixing the system that kept wasting it.

EPILOGUE

The hardware supply problem that never truly was.

At the beginning, it looked like a hardware story.

That was the easy read.

The platform team had a cluster running out of room. Finance was staring at a multimillion-dollar CapEx ask just as memory and storage prices were spiking. AI demand was distorting the hardware market and lead times were stretching right when the business needed capacity.

So the obvious question was: How much more hardware do we need?

But that was precisely the wrong question.

The better question was: **Why do we need more hardware when the cluster is not effectively using the hardware we already have?**

That was the game changer.

The app teams were not villains. They were rational. They owned reliability, so they protected themselves and the company because they have long memories of all kinds of outages. So, they added buffer.

The platform team was not helpless. But it was trapped in a model where challenging every request required trust it had not yet built. They had been trying to solving the problem with solutions aimed at the app team, not built with them.

Finance was not the enemy. It was being asked to fund the only solution the organization had made easy.

Everyone was acting rationally. The cluster filled with perfectly logical waste. That was the problem because nobody recognized the real issue was trust…and no one owned the trust gap.

This is why forced automation is such a tempting but critical mistake.

Some vendors look at the dashboard and think the answer is obvious. CPU is low. Memory is low. Requests for both are high. Turn on full autonomous rightsizing. Let the machine fix it.

That shows a vast misunderstanding of enterprise IT and how Kubernetes actually works inside these organizations.

Large companies are not slow because they are stupid. They are slow because the blast radius is real. Ownership is fragmented, production scars last, and regulated workloads exist. Pager duty changes behavior, and every app team knows the difference between a good recommendation and a safe operating model.

The question is not whether automation can rightsize better than humans staring at spreadsheets. Often, it can. The question is whether the organization trusts automation enough to let it act. **And that trust is not granted by a product claim; it is earned through proven behavior.**

The answer is adaptive autonomy that starts with visibility, then recommendations, followed by approval flows with necessary guardrails in place. Then controlled action that leads to broader autonomy where confidence has been proven. It adapts to each team, each workload, each risk profile, and ultimately each company's speed of trust.

It does not make autonomy less ambitious.

It makes it survivable.

The lesson of the wasted cluster is that the hardware crunch did not create the problem.

It simply removed the hiding place. When hardware was cheaper and easier to get, enterprises could buy around bad behavior. Now that memory, storage, chips, GPUs, servers, power, and data center capacity are all under pressure, waste has become too expensive to ignore.

But the answer is not to panic-buy hardware. And it is not to force automation faster than trust can absorb.

The answer is to make utilization a **shared responsibility** and let automation adaptively earn more control over time using a solution specifically designed for the enterprise model.

BEFORE THE NEXT EXCEPTION

Before you ask Finance for another hardware exception, ask a harder question:

Has your organization earned the right to use the capacity it already owns?

Because the cluster may not be full. It may just be waiting for automation to mature and be adaptive enough to **move at your speed of trust.**

ABOUT CLOUDBOLT

Automation that moves at your speed of trust.

CloudBolt Software helps enterprises turn cloud complexity into operational advantage by bringing intelligent automation, governance, and optimization to public, private, hybrid, and multi-cloud environments.

Its portfolio combines industry-leading cloud management with AI/ML driven Kubernetes rightsizing and optimization, enabling organizations to actively control and optimize cloud usage as it happens. CloudBolt empowers platform engineering, IT operations, and finance teams with the insight and automation they need to run cloud environments at scale, efficiently, continuously, and with confidence.

Learn more