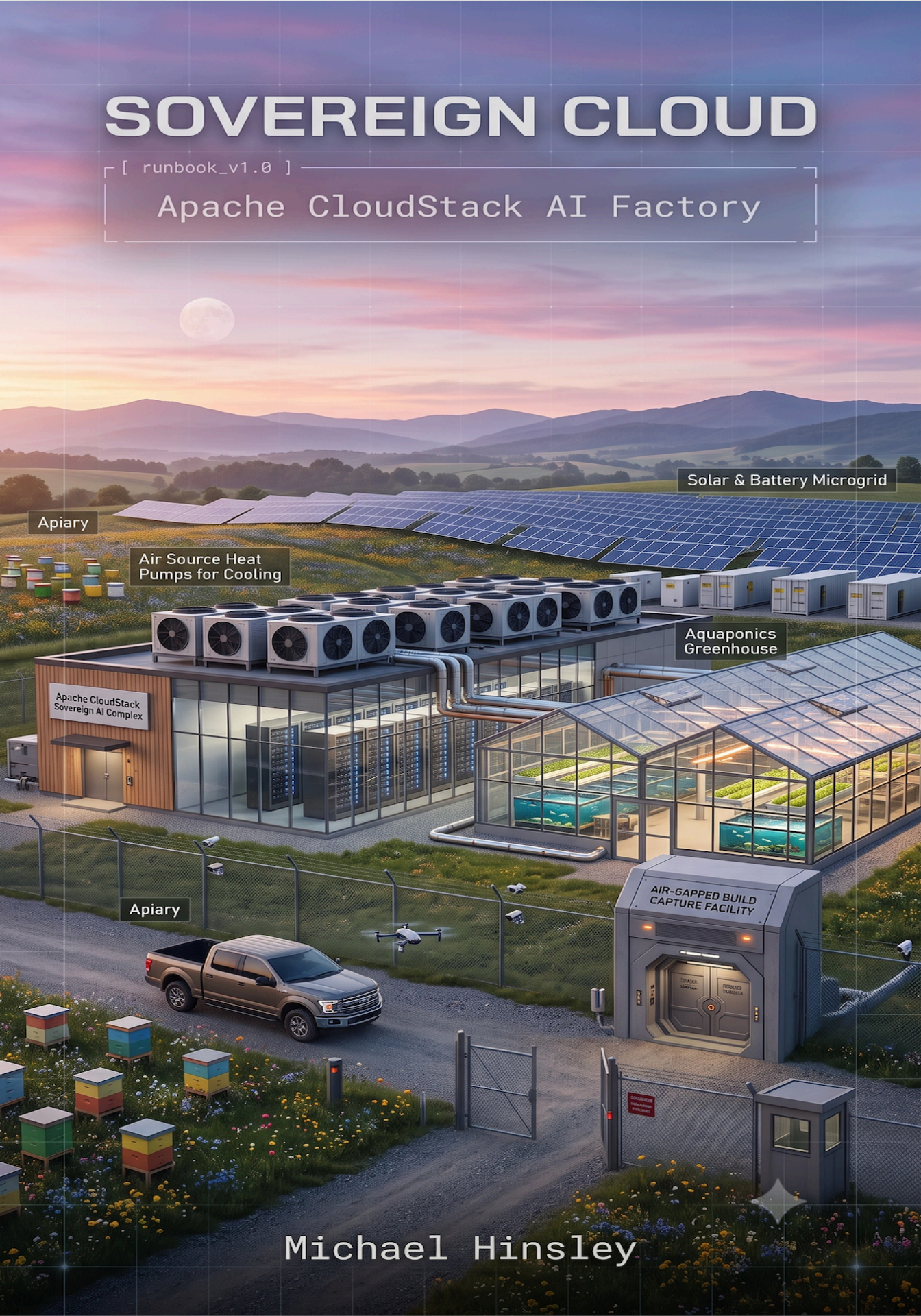
SOVEREIGN CLOUD

[runbook_v1.0]

Apache CloudStack AI Factory



Michael Hinsley

Apache CloudStack — AI Factory Edition

Building and running a flexible, small-scale yet powerful local AI factory on commodity enterprise hardware

Michael Hinsley

This book is available at https://leanpub.com/apache-cloudstack-ai-factory-edition

This version was published on 2026-08-07

Contents

Introduction — Read This First

There is a particular kind of letter that changes how an engineer thinks. For some it is a licensing renewal with a number on it that turns a tool you depended on into a threat. For a growing number of us, lately, it has been a different letter altogether: the notice that a model you had built into your product is being withdrawn, repriced, or placed behind an export control you cannot satisfy. On the twelfth of June 2026, two of the most capable models in the world were pulled from broad access at a stroke. If your business sat on top of them, that was not a news item. That was a Tuesday you will remember.

This book is the long answer to that letter. It builds, on hardware you can buy today and rack this week, a **local AI factory** — a small, owned, sovereign platform that runs frontier-class open models on your own iron, serves them to more than one user or tenant at once, and puts them to work earning their keep. No per-token meter. No revocation risk. No data leaving the building. When you own the stack, "can we run this?" stops being a vendor's decision and becomes an engineering one — and that shift is the whole point of what follows.

What this book is

It is a hands-on build guide, in the literal sense: you follow along on real Dell iron in a real rack drawing real watts, and at the end you have a working thing. The platform is **Apache CloudStack** — the same open cloud orchestrator that runs serious estates elsewhere — bent to a new purpose. Two Dell R7525 servers become CloudStack/KVM compute modules with GPUs and 100Gb RDMA passed straight through into virtual machines. Two R740XD boxes with 768GB of RAM apiece hold a frontier-class open model resident in memory and answer hard questions. An Ampere Ultra workstation runs an ARM model farm. A management node — an R660, or two Raspberry Pi 5s if you want to prove the point that enterprise iron is not mandatory at every layer — ties it together.

From those same machines you get the centrepiece: **the flex**. The two R7525s redeploy, from saved templates, as a four-node inference cluster, or a two-node aggregated cluster, or four isolated single-tenant boxes — three different machines from one pair of servers, switched by destroying and redeploying, never by pretending a passthrough VM can live-migrate when it cannot. The templates are the backups. That is a sovereignty story you can hand to an auditor.

Then the book puts the factory to work. You build **retrieval over your own documents** so the system answers from what you actually know. You **fine-tune small models** to sound like you and follow your formats. You wire the tiers together with an **n8n multi-agent support desk** — triage, answer, guardrail, escalate to the big brain only for the hard cases — and you stand up an **AI-first web and mobile front-end** built by a coding model running on your own endpoint. Every one of those outputs runs on hardware you own, with nothing crossing the boundary of your building.

Who it is for

The reader I have most in mind is the engineer inside an organisation whose data legally cannot leave its walls — financial services, legal, healthcare, government, anyone living under serious data

governance. For you, the public AI providers are not merely an expense; they are a compliance problem you cannot make go away with a contract clause. A factory you own is the answer that survives an audit.

But the book is also for the engineer who simply wants to own what they run — who has watched rented infrastructure become a leash and decided to take it back. You do not need a data centre. You need a rack, some commodity enterprise hardware, the patience to build it in a lab first, and the discipline to measure rather than assume. If you have those, you can build everything in these pages.

You do not need to be an AI researcher. You need to be comfortable on a Linux command line, willing to read a manual, and honest with yourself about what works and what does not. The hard-won specialist knowledge — what a mixture-of-experts model actually demands of a memory bus, why a consumer GPU will only do passthrough and never slicing, where RDMA over a back-to-back link stops being a clean mesh — is taught where it is needed, grounded in the hardware, and never hand-waved.

The thesis: you do not need to be a hyperscaler

The argument running underneath every chapter is simple, and the book holds it honestly in both directions. **You do not need to be a hyperscaler to own a capable AI factory** — a small one, built deliberately, can serve real models to real users with real power, and for an organisation under data governance it can do something the hyperscalers structurally cannot: keep the data home. The open-weight models that make this possible are genuinely good now, genuinely free to self-host, and — crucially — cannot be revoked out from under you once the weights are on your disks.

And the honest other half: **you should not pretend you are a hyperscaler when you are not.** There are things a hyperscaler does that this factory does not — elastic burst to thousands of GPUs, frontier-scale training, global low-latency presence. The book names those limits plainly rather than selling past them. Every figure in these pages is either timeless engineering, grounded and dated at the time of writing, or flagged for you to verify on your own build. Where a number is a vendor's claim, it is labelled as a vendor's claim. The brand of this book is honesty, and a single confident-but-wrong figure would undo it.

How to use this book

Read it in order. It is built like the platform it describes: each chapter stands on the one beneath it, and skipping ahead will leave you standing on a floor that is not there yet. The parts move from hardware, to the flex that makes the hardware versatile, to the brains that run on it, to the commercial outputs that make it pay.

Three habits will serve you throughout, and they are the book's own working discipline:

- **Build it in a lab and break it before it matters.** Every destructive step — and the flex itself is destructive by design — is rehearsed somewhere it costs you an evening, not a career.
- **Measure, do not assume.** The throughput of a giant model on a memory-bound box, the real recall of your retrieval pipeline, the deflection rate of your support desk: these are numbers you take on your own estate, not numbers you inherit from a tutorial. The book gives you the method; your hardware gives you the answer.

- **Apply the Mars test.** If this node were on Mars and you could not drive out to it, could you still recover it? If the answer is no, the design is wrong. You may one day run this factory lights-out for a regulated client, and that is exactly the standard it must meet.

There is one more thread worth naming up front: **Return on Research**. A pool of consumer GPUs and a pair of big-memory boxes are finite. Every watt, every gigabyte of VRAM, every core has to earn its place. The book spends them deliberately and asks you to do the same.

The shape of the journey

The build runs across four parts and twenty-two chapters. Here is the arc, so you can see the whole shape before you start; a full annotated index of every chapter sits at the front of the published edition.

Part I — Build

The foundations. Why owning the factory matters and what sovereignty actually buys a regulated firm; the honest bill of materials across the four tiers; racking, power, and the off-grid interlock that treats energy as a first-class design constraint; the network fabric — a quad 25Gb front-end and the back-to-back 100Gb RDMA link — and a clear line drawn around where CloudStack's responsibility ends; and the management plane, built two ways, with the single-node honesty stated rather than hidden.

Part II — The Flex

The heart of the versatility. CloudStack stood up on the R7525s with the GPU and the NIC treated as first-class, passed-through resources; then the flex itself — Profiles A, B and C as saved templates, the chapter that sells the book — and the RDMA inference fabric, with RoCEv2 reality-checked rather than oversold and vLLM stretched across the cluster for exactly what the back-to-back wiring can and cannot deliver.

Part III — The Brains

What runs on the platform. The quality brain — a frontier-class open model held in 768GB of RAM via llama.cpp, with its honest tokens-per-second envelope on real DDR4; the ARM model farm on the Ampere, CPU-or-GPU; retrieval over your own corpus, so the factory knows your documents; and fine-tuning small models with LoRA and QLoRA on a single consumer GPU, with the hard limits — you do not full-fine-tune the giants here — stated plainly.

Part IV — Make it Pay

The worked commercial outputs, and the departments that spend them. The data estate counted first — a bill of materials for what the organisation knows, before a further pound is spent on models; multi-agent orchestration with n8n; the support-desk pipeline that is the book's capstone — triage to a fine-tuned small model and retrieval for the answer, a guardrail check, and escalation to the big local brain only when it is earned; markdown turned into an AI-first web and mobile front-end by your own coding model; the factory walked into the departments — the engineers first, then editorial,

authors and production, then the back office and leadership, each workload carrying its guardrail and its honest demonstrated-versus-pattern ledger; the multi-tenant and data-governance chapter that makes the regulated-firm case concrete; operating the factory by the Mars test, with monitoring and a power-and-thermal budget; and finally the honest horizon — the edge, scale-out, and the limits where a hyperscaler genuinely wins.

A method, not a product

What you will hold at the end is a method, not a product, and not a guarantee. The architecture and the figures are illustrative, grounded in one specific build; your numbers will differ, and the discipline — measure before you change, rehearse before you commit, keep the evidence — is the real deliverable, because it travels to any infrastructure you ever touch.

The models in these pages will age. A better open model will ship the month after this is published, and the specific tokens-per-second figures will be overtaken. That is fine. The factory does not. The architecture, the flex, the way the tiers fit together, and above all the principle underneath them — **grow your own, cook your own** — outlast any single model. Build it, break it in a lab, and make it yours.

Now let us take the infrastructure back.

* * *

Apache CloudStack — The Sovereign AI Factory

Building and running a flexible, small-scale yet powerful local AI factory on commodity enterprise hardware

Leaf Spine Books edition

* * *

Use of code and examples. This book exists to help you do your job. The code, configuration, and command examples are provided for you to use and adapt in your own deployments – you do not need to ask permission to put them to work, and attribution, while appreciated, is not required. What does require permission is reproducing a substantial portion of the *text*, or redistributing the book itself. The method is yours to take; the words are not.

Disclaimer of warranty. This book is provided on an "as is" basis. While every effort has been made to ensure the accuracy of the information and the correctness of the procedures, the author and publisher make no warranty, express or implied, as to its completeness, accuracy, or fitness for any particular purpose. Open-weight models, hardware behaviour, and the surrounding software move quickly; a figure or a model that was current at the time of writing may have changed by the time you read this.

A working book on real infrastructure. The procedures here operate on live systems. Many of the commands are powerful and some are destructive: they can erase data, interrupt services, redeploy a node from a template over the top of a running one, and render systems unbootable. The factory's flex is deliberately a destroy-and-redeploy operation, not a live migration – treat it as such. Always rehearse in an isolated lab or a non-production environment first, keep tested backups, and understand each step before you run it. You assume all risk for any action taken on the basis of this book. To the fullest extent permitted by law, the author and publisher accept no liability for any loss, damage, data loss, downtime, or cost arising directly or indirectly from the use of, or reliance on, the material in these pages.

A method, not a product – and not a guarantee. The architectures, model choices, throughput figures, memory footprints, and any costs in this book are illustrative and grounded in one specific build on one specific set of hardware. Where a figure is a measurement, a vendor-reported number, or an illustration, it is labelled as such. Your hardware, your models, your quantisation, your corpus, and your operating conditions will differ. Treat every number as a starting point to be re-measured against your own estate, never as a promise of outcome. Above all: you do not need to be a hyperscaler to own an AI factory – and you should not pretend you are one.

A note on the framing. To keep the engineering concrete, this book frames its worked commercial outputs around a regulated organisation – the kind whose data, under financial, legal, or health-sector governance, legally cannot leave its own walls – and uses placeholder tenants (referred to as Tenant A, Tenant B, and so on) to stand in for that organisation's internal teams or customers. These are illustrative devices, not real organisations; any resemblance to a specific business, customer, or event is coincidental. The Leaf Spine Books imprint under which this book is published is real.

A note on the story. To keep the engineering concrete, this book follows a fictional organisation – the traditional publisher Leaf Spine Books, whose story began in the companion deployment guide, where it escaped a punitive licensing renewal by building its own cloud. In this book it builds its own AI factory. The organisation, its multi-city estate, and its named staff, tenants and clients are narrative devices; the characters are inventions, not portraits. Any resemblance to real persons, organisations, or events is coincidental. (The Leaf Spine Books imprint under which this book is published is real; the in-story publisher that shares its name is part of the narrative.)

* * *

First edition – 2026. Covers Apache CloudStack 4.22.1.0 LTS. Set in British English.

Leaf Spine Books Published in the United Kingdom.

* * *

On owning what you run

A word on the spirit of this book. Its philosophy is **GYOCYO – *Grow Your Own, Cook Your Own***: the conviction that if you cannot control your infrastructure, you do not truly own your business – and that this is more true of artificial intelligence, not less. A model you rent can be repriced, throttled, or revoked out from under you; one you hold on your own iron answers only to you. The notices

above protect the words on these pages. The method inside them is meant to be taken, adapted, and run on hardware you own — built, broken in a lab, and made yours. That is the whole point.

* * *

The Book in Full — Annotated Contents

Apache CloudStack — The Sovereign AI Factory **· Leaf Spine Books**

This is the whole book at a glance: every chapter, what it builds, and how the parts stack into a working AI factory you own outright. The build runs in order — hardware, then the flex that makes the hardware versatile, then the brains that run on it, then the commercial outputs that make it pay. It is written for the engineer whose data cannot leave the building, and for anyone who would rather own their infrastructure than rent it. Read it in sequence; each chapter stands on the one beneath it.

* * *

Part I — Build

1 · **Why Own an AI Factory** - *Sovereignty, the regulated-firm case, and the export-control reality.* A model you rent can be repriced, throttled, or revoked — and for a firm under data governance that is a compliance problem no contract clause can fix. This chapter sets the thesis the rest of the book defends in both directions: you do not need to be a hyperscaler to own a capable AI factory, and you should not pretend to be one. It also introduces Return on Research — the discipline that every watt and every gigabyte must earn its place.

2 · **The Build of Materials** *The four tiers, and an honest bill of materials.* The two flex servers, the two big-memory quality brains, the ARM model farm, and the management node — much of it commodity, refurbished enterprise iron, bought because it delivers the most capability per pound. What each tier is for, and why consumer GPUs earn their place over costing many times more.

3 · **Racking, Power and the Off-Grid Interlock Energy as a first-class design constraint.* Racking the estate and treating power — including an off-grid interlock — as something you design for rather than assume. Handled as method: a budget you measure on your own supply, never a figure borrowed from a book.

4 · **Network Fabric** A *quad 25Gb front-end, a back-to-back 100Gb RDMA link, and where CloudStack's responsibility ends.* The two networks the factory runs on, with a clear line drawn around the switching the core build genuinely requires — and a scoping note pointing to the wider switch estate and network-OS layer for those who need it.

5 · **The Management Plane** *Built two ways: a Dell R660, or two Raspberry Pi 5s, side by side.* The control plane that ties the estate together, shown on datacentre-aligned iron and on low-power hardware to prove enterprise kit is not mandatory at every layer — with the single-node no-HA reality stated plainly and the HA path named.

* * *

Part II — The Flex

* * *

Part III — The Brains

* * *

Part IV — Make It Pay

the head-out-of-the-sand first step for the non-technical reader, and it seats every departmental chapter that follows. The Part IV opener.

14 · **n8n Multi-Agent Orchestration** *Wiring the tiers into agent workflows.* The secure-by-default automation engine that becomes the glue for the commercial outputs, calling the factory's brains over their own OpenAI-compatible endpoints.

15 · **The Support-Desk Pipeline** *The capstone: triage, retrieve-and-answer, guardrail, escalate.* A multi-agent support desk that handles volume on cheap small models, grounds its answers in retrieval, checks itself at a guardrail, and escalates only the hard cases to the big local brain — with nothing leaving the building and an audit trail built onto every ticket.

16 · **Markdown to an AI-First Web/Mobile Front-End** *Built by your own local coding brain.* Turning your existing markdown into a static site and an installable progressive web app using a local coding model, then embedding an assistant that calls back into the factory's own retrieval and support-desk pipeline — the product consuming the brains that built it.

17 · **The Engineers** *The builders use the factory first.* The platform team's ask-the-estate retrieval over its own runbooks and post-incident records; the network engineers' explain-this-diff; security's strongest cannot-leave-the-building case; and the developers' local coding endpoint — each walked through the same fixed template, with the guardrail on what the AI must not do stated before the first user, and everything honestly labelled pattern until a measurement closes it.

18 · **Editorial, Authors and Production** *The promise the house can sign.* The author trust pitch — your manuscript never goes to a third party, never trains anyone's model, never leaves our building — assembled from standing machinery into a signable clause; consistency-checking never ghost-writing, with the line named and enforced as architecture; and metadata, accessibility and the backlist as both corpus and revenue.

19 · **Back Office and Leadership** *Where the pattern meets the law.* HR with the AI-never-decides-hiring guardrail stated plainly; finance where the AI never moves money and never adjusts a royalty; legal and rights, including subject access as the job that requires sovereign AI; and the leadership's own governance — because you cannot audit a black box you rent, and the estate's records are the audit.

20 · **Multi-Tenant and Data Governance** *Isolation, and the regulated-firm value-add.* Keeping tenants apart across compute, vectors, tickets, object storage, and secrets — and the hard truth that "compliant" is not a switch but an organisational programme, to which the factory contributes a sovereign architecture and a demonstrable evidence trail.

21 · **Operating the Factory** *Monitoring, the Mars test, and a power-and-thermal budget.* Running the estate by the rule that you must be able to recover a node you cannot physically reach, with honest observability on consumer hardware — including which telemetry a consumer GPU simply will not give you.

22 · **Where Next** *The edge, scale-out, and the honest limits.* The horizon: quantising down toward the edge, scaling out through a switch into a true N-way fabric, and a clear-eyed account of where a hyperscaler genuinely wins — closing the loop back to where the book began.

* * *

Front and back matter

- **Introduction** — what the book is, who it is for, and how to use it.

- **Copyright and edition notice** — the licence, the trademarks, and the GYOCYO close.
- **Glossary** — the working vocabulary, technical and coined, in one place.
- **Appendix A — Bill of Materials** — the four-tier reference build, part by part.
- **Afterword** — on owning what you run, with a bounded note on the author.

* * *

The expanded edition

This *is* the expanded edition, assembled: twenty-two chapters, built and verified. The factory's story joins the Leaf Spine Books adventure begun in the companion CloudStack deployment guide — short story sections framing each chapter with Sarah Martinez's team, wrapped around the same engineering honesty, with nothing removed. Four chapters are new to this edition, all in Part IV: the Data Estate opener, the Engineers, Editorial, Authors and Production, and Back Office and Leadership. Buyers ride Leanpub's free-update model; the work ships when it is verified, which is the only schedule this imprint keeps.

* * *

Releasing under the Leaf Spine Books imprint. The factory is ordinary hardware and open weights; what makes it yours is the method — built, broken in a lab, and run on iron you own. Grow your own, cook your own.

* * *

Chapter 1 — Why Own an AI Factory

Walk into the room where this book lives and the first thing you meet is the noise. Not the abstract hum of "the cloud" – somebody else's building, somebody else's meter – but a real rack on a real floor, pulling real watts you can read off a display. Fans. The faint tick of spinning metal and the steadier draw of silicon under load. A handful of Dell servers, a few consumer graphics cards that cost less than a second-hand car between them, two short cables running back-to-back between a pair of machines, and a management node you could lose behind a monitor. That is the whole factory. It fits in a corner. It belongs to you.

This chapter does not build any of it. It makes the argument for why you would want to, and it sets the terms of the rest of the book – including the terms under which I will be honest with you when something is harder, slower, or more lab-grade than the marketing around this subject would have you believe. We will get to the building soon enough. First, the why.

A word before we start, about the story that runs through this book. Readers of the companion deployment guide will recognise the device: to keep the engineering concrete, that book followed a fictional traditional publisher – Leaf Spine Books – through a real migration, out of a punitive licensing renewal and onto a cloud of its own. I am keeping the fiction running here, because it worked. The publisher, its staff and its three-city estate are inventions; the hardware, the commands, the caveats and every VERIFY around them are not. The story wraps the honesty, never the other way round. Here is where it resumes.

The Story: the second letter

Three weeks after the estate review closed, the second letter arrived.

Sarah Martinez recognised it for what it was before she had finished reading it, because she had kept the first one. That letter – the renewal quote with the 567 per cent increase that had started everything – lived in the top drawer of her desk in Manchester, and she had re-read it the morning of the review while Elena Novak set up the cost figures next door. It had become a kind of talisman. It said, in effect: your infrastructure is rented, and the landlord has noticed.

The second letter was not addressed to Leaf Spine Books at all, and it was not about servers. It was a service notice, forwarded by Marcus Chen with a one-line note – "read this the way you read the renewal quote" – and it concerned the hosted frontier model that two of the pilot tools upstairs leaned on. Access for whole categories of users, suspended. Not for anything anyone had done; policy had moved, an export directive had landed, and a capability that had been there on Friday was gone by the following week. Somewhere in the same seven days, a model of comparable quality had been published on the open internet under a licence that fit on one page.

Sarah read the notice twice, then walked it down to the room where the estate lived on the wall – Priya Patel's dashboards, the three cities glowing their usual quiet green. The migration had given them this: their own cloud, on their own iron, in their own buildings, and a team that had proved it could run it. James Wilson's restore drills. The runbooks Elena kept. Tom Fletcher's pager, quiet for a respectable number of weeks now.

"The first letter told us our infrastructure was rented," she said, when the team had gathered. "This one says our intelligence is rented. Same landlord problem. Different floor of the building."

Marcus, predictably, wanted the running order. Priya, predictably, had the question – where does the data go while these rented models are thinking? – and nobody in the room enjoyed the answer. Tom asked the honest one: if a model they rented could be switched off from a capital city none of them lived in, what exactly had they been building on?

"On sand," Sarah said. She uncapped her pen, wrote one line on the whiteboard, and underlined it. *We own the building. We are going to own the brains in it.*

What that meant in iron, in models, in watts and in working pipelines – that is what the rest of this book is for. The story hands over to the engineer here, as it always did.

The thesis: you do not have to be a hyperscaler

The prevailing story about artificial intelligence is a story about scale, and the scale is meant to frighten you off. Tens of thousands of accelerators. Buildings with their own substations. Capital that only a handful of companies on earth can raise. The implied conclusion is that serious AI is something you rent, by the token, from one of those companies, forever.

I want to take that conclusion apart, because it is half true and half a sales pitch.

The half that is true: if you want to *train* a frontier model from scratch, yes, that is hyperscaler territory, and nothing in this book pretends otherwise. The half that is a sales pitch: the idea that *using* serious models – running them, serving them to your colleagues, putting them to work on your own documents and your own problems – also requires that scale. It does not. The open-weight models have closed the gap, the hardware to run them sits on the second-hand enterprise market at sane prices, and the software to orchestrate it is free and battle-tested. A small factory, on commodity enterprise iron, can serve more than one person and more than one model at the same time, with real power behind it.

That is the thesis of this book, and it is Mike's thesis before it is mine: **you do not need to be a hyperscaler to own an AI factory, and a small one can do genuine work.** Owned, not rented. On your floor, on your power, answering to you.

There is a layer under that thesis, and the events this chapter describes are what taught it to us. Call it the three-layer inversion: **the data appreciates, the framework persists, the model depreciates.** The model – any model, however capable – is a replaceable part: the revocation week described below proved it, when a frontier-class replacement for a withdrawn model arrived on the open internet inside the same seven days. The framework – the factory itself: the pipelines, the retrieval, the guardrails, the plumbing that turns a model into working systems – outlives every model that passes through it. And the data – your documents, your records, the accumulated knowledge of your organisation – is the only layer of the three that appreciates with age, and the only one a competitor cannot download. Sovereignty, properly understood, is not "own the model". It is: own the refinery, because the crude is yours. The model is the current catalyst, and catalysts get swapped.

"Factory" is a deliberate word. It is not a single clever box with a chat window bolted on. It is a small set of machines, each tuned to a different job, wired together so that work flows through them – questions in, answers out, with retrieval and routing and guardrails in between, the way a real production line moves a part from station to station. By the end of this book you will have built that line and put it to work.

What this factory is — and what it is not

Honesty starts with scope, so let me draw the box before I sell you what is inside it.

This factory is four tiers of hardware doing four kinds of work. There is a *responsive tier* – a pair of Dell R7525 compute servers that, through CloudStack, can be reshaped on demand into whichever cluster the moment calls for. There is a *quality-brain tier* – two large-memory R740XD machines holding a very capable open-weight model in ordinary system RAM and answering the hard questions slowly and well, the model running not on bare metal but inside a very large virtual machine that CloudStack schedules onto them like any other guest. (That a giant-memory model can live inside a CloudStack virtual machine at all – rather than demanding a bare-metal box of its own – is much of what Chapter 9 is about.) There is an *ARM model farm* – a single Ampere workstation running the odds and ends that do not fit neatly on the others. And there is a *management plane* – the small node that runs the show, which can be a modest Dell or, to make a point, a couple of Raspberry Pis. Six consumer graphics cards across the whole estate, ninety-six gigabytes of video memory all told. That is the factory. Chapter 2 lays out the full bill of materials without flinching.

A word on the ground it all stands on. The CloudStack cloud these machines run as – the zone, the pods, the networking and the storage underneath – is built in full in the companion CloudStack deployment guide from Leaf Spine Books, and this book does not re-lay that foundation. Here the four tiers join that existing platform as **three clusters in a single multi-architecture zone**: the two R7525s, the two R740XDs, and the ARM farm, x86 and ARM sitting side by side under one control plane (the management node is that control plane, not a fourth cluster). Where a step belongs to the base cloud rather than the AI factory standing on top of it, I will point you to that companion build instead of repeating it here.

Now what it is *not*, stated plainly so you are never misled:

It is **not** a model-training rig. The large models in this book are run for *inference* – asked questions, not taught from scratch. Where we do training, it is the small, surgical kind on small models, and I will be precise about that distinction when we reach it.

It is **not** built on data-centre accelerators. The graphics cards here are consumer Ada-generation cards. They earn their place, but they have hard limits – no enterprise partitioning tricks, passthrough only – and I will name those limits rather than paper over them.

It is **not** a magic switch. The factory's headline feature, the ability to become several different machines on demand, is a *deploy-and-destroy* trick built on templates, not a seamless live shuffle of running workloads. It is genuinely useful and genuinely owned. It is not sorcery, and I will not describe it as if it were.

And it is **not** a finished product you download. It is a *pattern* – a way of assembling commodity parts into something coherent – that you adapt to your own corpus, your own users, your own constraints. The value is in understanding it well enough to make it yours.

Hold me to all four of those. If at any point the book starts sounding like a brochure, you have my permission to be suspicious.

The strongest reason: data that cannot leave the building

People come to self-hosted AI for all sorts of reasons – curiosity, cost, the sheer satisfaction of owning the thing. Those are real. But the reason that turns this from a hobby into a serious proposition, the one that puts a business case on the table, is **data governance**.

There are organisations for whom sending data to a third party is not a preference to be weighed but a line that legally cannot be crossed. A law firm holding privileged client material. A clinic holding patient records. A finance house holding positions and client identities under a regulator's eye. For these organisations, "we'll send your documents to an API and the answers will come back" is a non-starter before the first benchmark is ever discussed. The data cannot leave the building. Full stop.

A hosted model, however good, asks you to send your data out to be processed. An owned model brings the processing to the data. That is the entire difference, and for a regulated firm it is the difference between *allowed* and *not allowed*. Everything else in this book — the speed, the flexibility, the worked examples — sits on top of that one structural fact: **nothing leaves the building.**

There are two quieter advantages that ride along with it, and they matter even if you are not regulated.

The first is that there is **no per-token meter running.** A hosted model bills you for every question and every answer, forever, and your most successful internal tool becomes your largest recurring bill precisely *because* it is successful. An owned factory has a capital cost and a power cost and then it is yours; using it more does not cost more per use. The economics invert. (I am not going to quote you a percentage saving here — the honest figure depends entirely on your workload, your power, and your hardware, and anyone who hands you a single headline number is selling something. We will work the real arithmetic where it belongs, against a real bill of materials.)

The second is **no revocation risk** — and that one has stopped being theoretical.

The live hook: the week a model could be pulled

I am writing this in June 2026, and I am going to date it deliberately, because the point only lands if you know it is current and not a hypothetical I invented.

This month, access to some of the most capable hosted frontier models was *suspended* for whole categories of users — not because anyone did anything wrong, but because policy moved. Export-control measures cut off foreign access to certain top-tier American models. One day a team had a frontier model underpinning their work; a policy directive later, they did not. The capability did not get worse. It simply became unavailable to them, by decision, from outside.

In the very same window, a Chinese lab — Z.ai, formerly Zhipu — released GLM-5.2: a mixture-of-experts model of genuinely frontier-competitive quality, with a context window of a million tokens, published on Hugging Face under an **MIT licence**. No usage restrictions. No regional locks. A free download you can pull onto your own machines and run until the disks wear out. (Re-checked as I write, 27 June 2026; the precise parameter count and the finer capabilities are a Chapter 9 matter, and I will hold the exact figures until we can stand them up on real hardware.)

Set those two events side by side, because together they make the whole argument for ownership in a single week:

> A model you *rent* can be taken away from you by someone else's decision. A model you *own* cannot.

That is not a slogan; it is a property of the arrangement. When the weights are on your disk under a permissive licence, no meter, no terms-of-service change, and no directive from a capital city you do not live in can switch your factory off. The capability is in the building, on iron you control. Whatever the politics do next, your line keeps running.

You do not have to take my word for any of that. The proof is one command long. The very model I just described publishes its licence as a plain file in its public repository, and you can pull that file onto your own disk right now — no account, no sign-up, no meter — and read the terms for yourself:

```
# Pull GLM-5.2's licence onto your own disk and read it — no account, no meter.
# The `hf` command ships with the huggingface_hub package.
pip install -U huggingface_hub
hf download zai-org/GLM-5.2 LICENSE --local-dir ./glm52-proof
cat ./glm52-proof/LICENSE        # -> the MIT licence, in full
```

That is sovereignty made tangible: a frontier-class model's terms, sitting in a folder you own, under a licence (MIT) that carries no regional lock and no remote off-switch. The weights themselves are a multi-hundred-gigabyte download we line up properly in Chapter 9 — the point here is only that *nothing stands between you and them but bandwidth.* (Grounded against the Hugging Face Hub CLI documentation and the `zai-org/GLM-5.2` model card, both read 30 June 2026; the exact CLI version and the download sizes are the kind of environment-specific detail this book always asks you to VERIFY on your own machine.)

This is the heart of the commercial case, and I want it stated without hype: ownership buys you **continuity**. For a regulated firm it also buys you *permission*. Everything else is gravy.

GYOCYO: Grow Your Own, Cook Your Own

There is an ethos under all of this, and it predates the AI part by a long way. I call it **GYOCYO — Grow Your Own, Cook Your Own.**

It is the same instinct that makes a person keep bees instead of buying honey, or run their own power instead of drawing it all from the grid: a preference for understanding and owning the means of a thing rather than consuming the thing as a finished product handed down from somewhere you cannot see. It is not nostalgia and it is not a rejection of buying things — you will buy plenty of hardware in this book. It is about where the *capability* lives. Grow your own: hold the means of production. Cook your own: do the work yourself, end to end, so you understand it.

Applied to AI, GYOCYO says: do not be only a consumer of intelligence sold by the cup. Hold the model. Hold the data it works on. Hold the pipeline that joins them. You will know more, depend on less, and own what you build. The factory in this book is GYOCYO made concrete in a rack.

Return on Research

The accountant's question is Return on Investment: what did the money buy. It is a fair question and we will answer it. But it is the wrong *first* question for a project like this, and if you let it be the first question you will talk yourself out of every worthwhile thing you could learn.

The first question is **Return on Research** — RoR. What did *building it* buy you, beyond the running artefact at the end?

When you stand this factory up yourself, you come away knowing how CloudStack carves real hardware into flexible virtual machines; how a large model actually behaves when it is held in system memory instead of a hyperscaler's accelerators; what RDMA does and does not give you on consumer

cards; how retrieval and fine-tuning differ and when each is the right tool; how to wire several models into a pipeline that routes, answers, checks itself, and escalates. None of that knowledge can be revoked, metered, or priced per token. It compounds. The next system you build, and the one after, are cheaper and faster *because you did this one the hard way.*

Return on Research is the honest reason a busy professional should build rather than buy even when buying looks cheaper on a spreadsheet. The spreadsheet cannot price what you will know. This book is, more than anything, an instrument for collecting that return.

The Mars heuristic

I keep one blunt test on the workbench for every design decision in this book, and I will hold you to it as much as myself. I call it the **Mars** test.

Imagine the factory is on Mars. The link home is hours of latency at best and cut entirely at worst. There is no calling out to an API, no fetching a model on demand, no support engineer on the other end of a chat window. Whatever you need, you needed to have brought with you, on your own iron, working offline.

Ask of any part of the design: *would this still run on Mars?* If the answer is no — if it secretly depends on a service you do not control, a model that lives on someone else's server, a licence that phones home — then it is not really *yours*, and it fails the test. The Mars heuristic is sovereignty stated as an engineering constraint. It is why the models in this book are downloaded and held, not called; why the secrets live in a vault on the management node, not in a cloud; why "it works as long as the internet is up and the vendor is happy" is not good enough. Cut the umbilical, and the factory should keep working. We return to this test directly in Chapter 21, when we operate the thing for real.

The map of the journey ahead

The rest of the book builds the factory and then puts it to work. Here is the shape of it, so you can see where each piece is going. I am sketching the route, not walking it yet — every claim below is built and reality-checked in its own chapter.

Part I — Build. Chapter 2 sets out the four tiers as a real bill of materials. Chapter 3 racks them, feeds them power, and meets the off-grid interlock head-on — running a factory partly or wholly on power you generate yourself. Chapter 4 lays the network fabric: the everyday front-end links, and the two short back-to-back cables that carry the high-speed RDMA traffic between the pair of compute hosts. Chapter 5 builds the management plane *twice* — once on a modest Dell, once on a couple of Raspberry Pis — to prove that enterprise iron is not mandatory at every layer.

Part II — The Flex. Chapter 6 stands CloudStack up on the two R7525s, with graphics cards and network cards handed straight through to the virtual machines as first-class resources. Chapter 7 is the chapter that sells the book: **the Flex.** The same two physical servers become three different machines — a tightly-coupled cluster, an aggregated pair, or four independent tenants — by deploying from templates and destroying what came before. The templates *are* the backups. I will be straight with you up front: this is deploy-and-redeploy, not live migration, and a virtual machine with hardware passed through to it cannot be migrated live at all. That constraint is not a footnote; it shapes the whole design, and Chapter 8 reality-checks the high-speed fabric on top of it — including the honest truth that two cables wired back-to-back give you cross-host *pairs*, not a clean four-way mesh.

Part III – The Brains. Chapter 9 stands up the quality brain: a large open-weight model held in the big-memory machines – run as a large-memory virtual machine under CloudStack rather than on bare metal – and answering carefully. Chapter 10 brings up the ARM farm for everything that does not fit elsewhere. Chapter 11 is **RAG** – retrieval over your own documents – which I will define carefully as the thing that makes a model *know your material*. Chapter 12 is **fine-tuning**, the small and surgical kind, which is a different thing entirely: it makes a small model *sound like you and follow your format*. Knowing your documents and sounding like you are two different jobs done two different ways, and confusing them is the most common mistake in this whole field. I will keep them firmly apart.

Part IV – Make It Pay. Chapter 13 counts the data estate – a bill of materials for what the organisation knows, before a further pound is spent. Chapter 14 wires the tiers together into orchestrated workflows. Chapter 15 is the capstone: a self-hosted **support desk** that takes a question, routes it, answers it from your own documents with a fine-tuned small model, checks the answer, and escalates the genuinely hard cases to the big brain – with nothing ever leaving the building. Chapter 16 takes your written content and stands up an AI-first front-end for it using a local coding model; I will be candid that this is a build *pattern* needing real iteration, not a one-shot app, and that "mobile" here means a web app in a thin shell, not an app-store production. Chapters 17 to 19 take the working machinery into the departments – the engineers; editorial, authors and production; and the back office and leadership. Chapter 20 makes the multi-tenant and data-governance case in full – the commercial heart. Chapter 21 operates the factory: monitoring, power and heat, and the Mars test applied for real. Chapter 22 looks at what comes next, honestly, including where this approach runs out of road.

That is the journey. Build it, then make it earn its keep.

The honesty contract

Before we start, a contract – my side of it.

This subject is drowning in hype, and some of the source material that informed this book was written in exactly that register: big round percentages, "bare-metal" performance claims on consumer cards, breathless figures with no working behind them. None of that survives into these pages. Where I give you a number, it will be one we can stand up and measure, or it will be flagged as something to verify on your own live build before you trust it. You will see the word **VERIFY** in this book, and it is not hedging – it is me marking, precisely, the boundary between what I can promise on paper and what only your own hardware can prove.

So, the contract:

- Where the approach is **lab-grade** – consumer graphics cards standing in for data-centre kit, RDMA on consumer network cards, a trillion-parameter-class model held on a single large-memory box – I will say so, in those words.
- I will not quote you an unbacked figure to make a point. No invented savings percentages, no "sub-microsecond" promises on consumer hardware, no licence or cost numbers I cannot show you the source of. Where a benchmark comes from a vendor, I will tell you it is the vendor's number.
- I will keep the model honesty tight: which models can see images and which cannot, which are general-purpose and which are coding-only, which jobs are inference and which are training, and where a capability is real versus where it is aspirational. The exact figures live in the chapters that can test them.

- And I will keep the personal and the private out of the worked examples. The contact details and the colophon live at the back of the book, and they are illustrative; the build pages deal in roles and patterns, not anyone's real address, accounts, or readings.

In exchange, I will give you the genuine article: a real factory, on real iron, that does real work and owes nothing to anyone once it is built. That is a fair trade, and it is the only kind this book is interested in making.

Turn the page. Let us count the iron.

* * *

Chapter 2 — The Build of Materials

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

The Story: counting what the building already holds

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

How to read this bill of materials

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

Tier 1 — The responsive tier: two Dell R7525s

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

Tier 2 — The quality brain: two Dell R740XDs, 768GB each

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

Tier 3 — The ARM model farm: one Ampere Ultra

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

The graphics-card pool, counted honestly

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

Tier 4 — The management plane: an R660, or a pair of Raspberry Pis

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

Where these four tiers live: three clusters in one zone

Proving the iron: a hardware-readiness pass

What is not on the headline list — but you still need

The shape of the cost — without a figure

Chapter 3 — Racking, Power and the Off-Grid Interlock

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

The Story: four of a kind

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

The rack as a machine with a shape

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

Power is not a wall socket

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

Measure the draw: a power-and-thermal readiness pass

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

The factory that parks itself

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

The off-grid interlock

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

Every watt on purpose

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

Chapter 4 — Network Fabric: the Quad-25Gb Front-End, the Back-to-Back 100Gb, and Where CloudStack Ends

The Story: the question on the wall

Two fabrics, one estate

The quad-25Gb front-end: the network CloudStack governs

The switch estate: what belongs in this book

The back-to-back 100Gb: short, switchless, and honest about its shape

Where CloudStack ends

Bring up the fabric: a network-readiness pass

The quad-25Gb front-end — five checks

The back-to-back 100Gb — four checks, first look only

Every wire on purpose

Chapter 5 — The Management Plane: the Conductor of the Estate, Built Two Honest Ways

The Story: the conductor they already had

What the conductor actually holds

The caveat, stated plainly: one node is no HA

Option A — the datacentre default: a Dell R660

Option B — the low-power showcase: two Raspberry Pi 5s

The HA aside: keepalived, HAProxy, and a replicated database

A note on management traffic, and a warning about the keys

Bring up the conductor — a management-node readiness pass

The Mars test, applied to the conductor

Built, not yet installed

Chapter 6 — CloudStack on the R7525s: KVM, and the GPU and NIC Made Schedulable

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

The Story: the hosts that waited

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

Installing against the conductor

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

KVM on the R7525s

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

A zone, a pod, a cluster

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

The multi-architecture zone, named and deferred

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

Making the iron schedulable

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

Host preparation: IOMMU and VFIO

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

The GPU as a first-class resource

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

The NIC, honestly: a delivered device, not a first-class resource

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

Why none of this can move

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

Bring the platform up — an install-and-passthrough readiness pass

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

The Mars test, applied to the install

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

An orchestra, ready to be told what to play

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

Chapter 7 — The Flex: Profiles A, B and C as Templates

The Story: the cluster that left

The template is the unit of the flex

Switching is destroy-and-redeploy, not live-migrate

Profile A — the four-node vLLM cluster

Profile B — the two-node aggregated pair

Profile C — four independent tenants

Placement is your job, not the scheduler's

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

The flex as code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

Build the flex — a profiles-and-templates readiness pass

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

Three machines, one rack

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

Chapter 8 — The RDMA Inference Fabric: RoCEv2 Reality-Checked, vLLM Across the Cluster

The Story: the number nobody would say

The link is RDMA, not merely a fast port

The switchless pair is the easy case — say so

The driver question: in-box first

GPUDirect RDMA: assume it is not there

vLLM across the pair: what distribution actually buys

The measured reality: a method, not a number

Build the fabric — an RDMA-and-vLLM readiness pass

The fast wire, reckoned with

Chapter 9 — The Quality Brain: GLM-5.2 (and Kimi) on the 768GB R740XDs via llama.cpp

The Story: the quiet join

The tier, and why it is a separate machine

Why this model: ownership you cannot have revoked

What GLM-5.2 actually is

The engine: llama.cpp, and the honest size problem

How llama.cpp spends the one GPU

Speed, told straight

The line-up: one brain, or two, and which ones

What this tier is for — and what it is emphatically not

Settled, and VERIFY-on-live

Build the brain — a quality-brain readiness pass

The brain that answers the hard ones

Chapter 10 — The ARM Model Farm: Ampere Ultra, CPU-or-GPU

The Story: the odd one in

The tier nobody else covers

Why ARM, on purpose

The multi-architecture zone, made real

CPU or GPU: the two ways this box serves

The engines that run clean on arm64

What this tier is for, and what it is not

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

Settled, and VERIFY-on-live

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

Build the farm — an ARM-farm readiness pass

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

The third tier, in its place

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

Chapter 11 — Corpus and RAG: Retrieval Over Your Own Documents

The Story: the one who asked what the data said

Two ways to make a model know something

The pipeline, end to end

Ingest and chunk: cutting the corpus to size

Embed: the second, smaller model

Store: Qdrant, the vector index

Retrieve and rerank: similarity is not relevance

Ground: which brain answers, and from what

What RAG is, and what it is not

The sovereign half

Build the pipeline — a RAG readiness pass

Settled, and VERIFY-on-live

The half that knows your documents

The Story: a promise with a number waiting

Afterword - The estate is yours now

The last chapter closed the build. This one steps back from it.

When this book began, the problem was a letter: a model withdrawn, a price changed, a capability you depended on placed somewhere you could no longer reach. Twenty-two chapters later, the answer is sitting in a rack you can put your hand on. It draws power you can meter, holds weights no vendor can recall, and answers questions without a single byte leaving the building. That is not a small thing. For an organisation that had been told its only options were to send its most sensitive data to someone else's computer or to do without, it is the difference between "no" and "yes".

It is worth being honest about what changed and what did not. The hardware in these pages is ordinary — last-generation enterprise servers and consumer GPUs, bought second-hand, racked in a room. The models are open weights anyone can download. None of it is exotic. What changed is not the parts but the posture: the decision to own the stack rather than rent it, and the discipline to build it properly — in a lab first, measured rather than assumed, recoverable from a cold start. The factory is the visible output. The discipline is the real one, and it is the part that travels. Hand these methods a different set of boxes, a better open model, a workload this book never imagined, and they still hold.

The thesis has been deliberately two-sided throughout, and it stays that way here. You do not need to be a hyperscaler to own a capable AI factory — that has, I hope, been demonstrated rather than merely asserted. And you should not pretend to be one. There are problems that genuinely want a thousand GPUs and a global footprint, and this book has tried to name them plainly each time one came into view, because a guide that oversells is worse than useless to the engineer who has to make it work on Monday morning. The honest boundary is part of the gift: knowing exactly what your factory does well lets you lean on it without fear, and knowing where it stops lets you reach for something else without shame.

There is a quieter point underneath all of it. Owning your infrastructure is not really about cost, or even about compliance, though it touches both. It is about which questions you get to ask. When the platform is someone else's, your imagination is bounded by their roadmap and their pricing and their terms of service. When it is yours, the only limits are physics, your skill, and your nerve. That is a larger freedom than it first appears, and it is the one most worth building for.

So take it and build. Break it in a lab where breaking is cheap. Measure everything. Keep the receipts. And when the next letter arrives — the model pulled, the price doubled, the door quietly closed — let it find you already standing on ground you own.

Grow your own. Cook your own.

* * *

About the author

Michael John Leslie Hinsley has spent over three decades bridging the gap between delicate electronics and the mud and rain of the physical world. From enterprise data centres to off-grid aquaponics and apiaries, his engineering philosophy remains the same: if you cannot control

your infrastructure, you do not own your business. This book turns that conviction on artificial intelligence — the most rented, most revocable layer of the modern stack, and for that reason the one most worth owning.

* * *

Stay in touch

This book is published under the Leaf Spine Books imprint, whose subject is sovereign, owner-operated infrastructure — built, broken in a lab, and made yours.

To follow the imprint, find its other titles, or get in touch about the method in these pages, reach Leaf Spine Books at:

Leaf Spine Books publishes at https://www.leafspinebooks.com/ — where the rest of the line appears, including *Apache CloudStack — A Production Deployment Guide*, whose estate this book's lab was proven on.

Published under the **Leaf Spine Books** imprint.

If the book helped you take back a piece of your infrastructure, that is the only signal it was looking for.

* * *

Glossary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

Appendix A — Bill of Materials

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

The four tiers at a glance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

Tier 1 — Responsive / flex compute: Dell PowerEdge R7525 (×2)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

Tier 2 — Quality brain: Dell PowerEdge R740XD (×2)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

Tier 3 — ARM model farm: Ampere Ultra workstation (×1)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

Tier 4 — Management: two documented options

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

The RDMA fabric

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.

On power and cooling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/apache-cloudstack-ai-factory-edition.