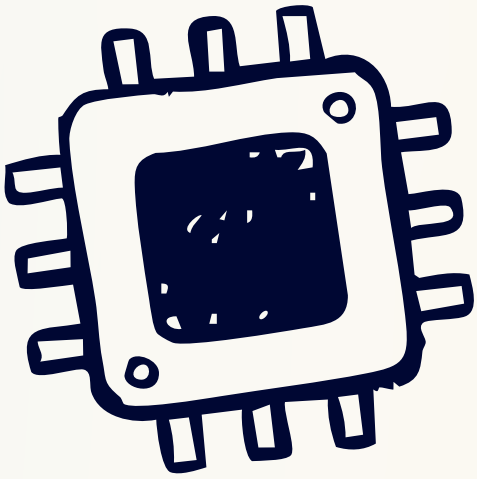
Oasium AI

GUIDE № 02



Local AI: a hardware buyer's guide for 2026.

Research labs, healthcare and government teams, sovereignty-focused enterprises, solo operators, and personal AI builders considering whether to deploy AI on hardware they own. Useful as a procurement worksheet or as a structured way to evaluate the local-vs-cloud question.

OPERATIONS · 32 PAGES · PDF

MAY 2026 EDITION

CONTENTS

What's inside

01

Why this guide exists

Eighteen months ago, the honest answer to "should we run AI locally?" was almost always *no*. The frontier APIs were dramatically better than anything you could run on hardware you owned. Open-weight models lagged closed models by twelve to eighteen months, and the gap was visible on most workloads. Hardware capable of running 70-billion-parameter models was either prohibitively expensive (eight-GPU server racks) or available only in specialized configurations few teams could justify.

In 2026 the situation has changed in three specific ways, all of them recent:

The open-weight models have caught up on most production workloads. Llama 4 Maverick scores 85.5% on MMLU; DeepSeek V4 scores 83.7% on SWE-bench Verified and 99.4% on AIME 2026; Qwen 3 235B leads reasoning benchmarks at 77.2% on GPQA Diamond. These are not toy benchmarks. They are the same benchmarks closed APIs are measured against. The closed frontier still leads on a few specific things — multi-hour goal-driven agents, latest-quarter reasoning capability, certain multimodal tasks — but the gap on bread-and-butter enterprise work (summarization, classification, structured extraction, code completion, internal Q&A) has effectively closed.

The hardware to run these models has gotten dramatically more accessible. A Mac Studio with M3 Ultra and 512GB of unified memory runs 70-billion-parameter models at usable speeds for under $11,000. NVIDIA's DGX Spark — released October 2025, currently $4,699 — runs models up to 200 billion parameters in a desktop form factor at 128GB unified memory. Dell shipped the Pro Max with NVIDIA's GB300 Grace Blackwell Ultra Superchip in March 2026: 748GB of coherent memory and 20 petaFLOPS at FP4 in a desk-side machine that replaces what used to be a small server room.

The software stack has matured. Apple's MLX framework makes Apple Silicon a first-class platform for ML inference. NVIDIA's NemoClaw and OpenShell, released March 2026, provide a sandboxed runtime for always-on local agents. Ollama, llama.cpp, and vLLM have settled into reliable production-quality tooling.

This guide is for anyone evaluating whether to deploy AI on hardware they own. It is opinionated about which workloads benefit from local deployment, which don't, what hardware to buy at each tier, what models to run on it, and what operational reality looks like in month four. It is not a survey of every option; it is a working recommendation document, written by people who do this for a living and update their thinking quarterly.

02

What "local" actually means here

Two different things are commonly described as "local AI." They solve different problems, run on different hardware, and have entirely different cost structures. Almost every confused procurement conversation about local AI is confused at this distinction. Resolve it before reading further.

The first kind: local model weights. The neural network itself runs on hardware that the user or organization owns. Open-weight models — Llama 4, Qwen 3, DeepSeek V3.2, V4 — are downloaded from Hugging Face or another source, served via inference software (Ollama, vLLM, MLX, llama.cpp), and queried by applications that send prompts to the local server rather than to a third-party API. No data leaves the building. This is the version this guide is mostly about.

The second kind: a local agent using hosted models. The agent process — the program that reads files, controls a browser, calls tools, makes decisions across multiple steps — runs on the user's computer or organization's infrastructure. But the underlying intelligence (the LLM) is a frontier API call to Anthropic, OpenAI, Google, or another hosted provider. Examples: Claude Code, OpenAI's Codex Desktop with the Atlas browser, the open-source OpenCUA project, NVIDIA's NemoClaw running on top of OpenShell. The agent is local; the model usually is not.

These two architectures have almost nothing in common from a buying perspective:

	LOCAL MODEL WEIGHTS	LOCAL AGENT + HOSTED MODEL
What you buy	Hardware capable of running large models in memory	A laptop or workstation that can run an agent process
What you pay over time	One-time hardware cost; minimal recurring	Per-token API fees (variable, can be substantial)
Data flow	Data never leaves your network	Prompts and context flow to the model provider
Sovereignty / compliance	Full control	Subject to provider's policies, data residency, etc.
Quality ceiling	Best open-weight model you can run	Frontier closed models (typically the best available)
Maintenance burden	Real — operations, monitoring, model swaps	Minimal — agent runtime + API keys
Right for	Sensitive-data orgs, sovereignty-required, offline contexts, high-volume steady-state workloads	Most use cases where data control isn't binding

If your driving constraint is data sovereignty, you need the first kind. If your driving constraint is "I want an agent that lives on my machine and works with my files but I'd rather get the smartest model possible," you almost certainly want the second kind.

The rest of this guide is about the first kind.

03

What changed on the model side

This section is dated. The open-weight model landscape moves quickly enough that any specific recommendation here will need refreshing in roughly six months. The structural pattern, however — open-weight catching up on workload-by-workload basis, choosing per workload rather than picking one universal model — is durable.

The open-weight tier in 2026

Llama 4 Maverick (Meta, 2025) is the strongest general-purpose open-weight model for most enterprise work. It uses a sparse mixture-of-experts architecture: 17 billion parameters are active on any forward pass, despite roughly 400 billion total weights in memory. The result is frontier-grade general-purpose performance with inference cost that fits on a single high-spec workstation. On MMLU it scores 85.5%. On most general-purpose tasks, the gap to a closed-API frontier model is small enough not to be the binding constraint. Hardware footprint: ~400GB of memory to keep weights loaded, ~17GB active per forward pass.

Qwen 3 235B (Alibaba) leads on reasoning-heavy workloads. On GPQA Diamond — a graduate-level science benchmark designed to be Google-proof — it scores 77.2%. On the AIME math suite, 85.7%. For internal scheduling, multi-step legal analysis, technical Q&A over engineering documentation, and any workload that resembles "think carefully about this," Qwen 3 235B is the strongest open-weight option. Hardware footprint: full 235B in memory.

DeepSeek V3.2 / V4 changed the architectural landscape twice. V3.2 (late 2025) made ultra-long context usable on the open side. The Speciale variant matches Gemini 3 Pro on math and coding contests and earned gold-medal results in the 2025 IMO and IOI. V4 (April 2026) advanced this further, leading current open-weight benchmarks: 83.7% SWE-bench Verified, 99.4% AIME 2026, 92.8% MMLU-Pro. V4 is the new default recommendation for document-heavy on-prem workloads — research libraries, legal archives, compliance histories. Hardware footprint: ~1 trillion parameters; requires multiple H100 GPUs or a Dell GB300 to self-host.

Smaller models worth considering:

- **Qwen 3-35B-A3B** punches far above its weight: 73.4% SWE-bench Verified using only 3 billion active parameters. Excellent for cases where memory budget is constrained.
- **Mistral models** (Large 2, Small 3) remain strong choices for European deployments where Mistral's compliance posture matters.
- **Gemma 4** (Google) is the easiest open-weight model to deploy at small scale; weaker on the hardest reasoning tasks but reliable.

How to pick a model for your workload

The right open-weight model depends on the dominant pattern in your workload. Use this matrix as a starting point; verify against your held-out test set before committing.

WORKLOAD PATTERN	RECOMMENDED	WHY
Long-document summarization	DeepSeek V3.2 or V4	Built for ultra-long context; high quality on extraction
Reasoning-heavy (math, scheduling, multi-step analysis)	Qwen 3 235B	Leads reasoning benchmarks at this tier
General-purpose mixed workload	Llama 4 Maverick	Best balance of quality and hardware footprint
Code completion / engineering Q&A	DeepSeek V4 or Qwen 3 235B	Highest open-weight scores on SWE-bench
High-volume classification / extraction	Smaller fine-tuned model + Qwen 3-35B fallback	Don't pay full cost for routine work
Memory-constrained deployment (single workstation)	Qwen 3-35B-A3B or Llama 4 (MoE-routed)	Active params smaller than total params

Anti-pattern: picking one open-weight model and using it for all workloads regardless of fit. The closed-frontier providers can do this because they spent billions of dollars training general models. You shouldn't pretend the same is true of any single open-weight model.

04

What changed on the hardware side

The hardware landscape has three usable tiers in 2026: workstation-class, desktop server-class, and rack-scale. We focus on the first two — anyone needing rack-scale already has internal capability and isn't reading buyer's guides.

Workstation-class: Mac Studio M3 Ultra

Apple's Mac Studio (https://www.apple.com/shop/buy-mac/mac-studio) with M3 Ultra is the workstation-class baseline for local AI as of mid-2026. Configurations matter, so the table below shows what you actually get for what you actually pay.

CONFIGURATION	RAM	STORAGE	PRICE (APPROX)	RUNS
Base M3 Ultra	96GB unified	1TB	$3,999	30B-class models comfortably; 70B with quantization
Mid M3 Ultra	192GB unified	2TB	$5,599	70B-class models native; some 100B+ at 4-bit
High M3 Ultra	256GB unified	4TB	$7,599	100B+ models comfortably
Max M3 Ultra	512GB unified	8TB	$10,799	Llama 4 Maverick fully loaded; near everything open-weight

Apple discontinued the 512GB SKU briefly during the global memory shortage of early 2026 — confirm availability before ordering at that tier. For most teams, the mid (192GB) configuration is the sweet spot: it runs 70B-class models in 4-bit quantization with headroom for a meaningful KV cache.

What runs well on Mac Studio: Inference under MLX (Apple's machine-learning framework, equivalent in role to NVIDIA's CUDA stack). Quantized 70B-class models. Fine-tuning on small-to-medium datasets. Single-user development workloads.

What doesn't run well: Multi-user serving at scale (CUDA-only inference engines like vLLM are limited). Training from scratch (Apple Silicon is decent for inference, behind NVIDIA for training). Tasks that require distributed setup across multiple machines (Apple's UltraFusion handles two dies; doesn't extend to a cluster).

Coming later in 2026: Apple has signaled an M5 Ultra Mac Studio for mid-to-late 2026 (originally targeted for WWDC 2026 in June, possibly delayed to October due to memory supply). Expected specs: 36 CPU cores, 80 GPU cores, up to 256GB unified memory, ~1100 GB/s bandwidth. The M5 Ultra

brings significant inference speed improvements via dedicated matrix-multiply units in every GPU core. If you're not in a rush, waiting for M5 Ultra may be worth it; if you need capability now, M3 Ultra is excellent.

Workstation-class: NVIDIA DGX Spark

NVIDIA's DGX Spark (https://www.nvidia.com/en-us/products/workstations/dgx-spark/) is a desktop AI workstation built around the Grace Blackwell GB10 superchip. It shipped October 2025; price increased from $3,999 to $4,699 in February 2026 due to memory supply constraints. Specifications:

- 128GB of unified LPDDR5x memory — low-power DDR5 memory packaged on the same module as the chip, giving the model much faster access than separate RAM
- ~1 PFLOP at FP4 precision (4-bit floating-point — a compression scheme that lets large models fit into less memory without much loss in quality)
- ~1,000 TOPS of inference performance
- 20-core CPU (10 Cortex-X925 + 10 Cortex-A725 ARM cores)
- 4TB NVMe storage
- 150mm × 150mm form factor; 1.2kg

What it's good for: Running models up to ~200 billion parameters via inference. Fine-tuning models in the 30B-70B parameter range. Single-developer or small-team workloads. Running NVIDIA's NemoClaw / OpenShell stack for sandboxed local agents (the hardware ships ready-configured for it).

Tradeoff vs. Mac Studio: DGX Spark has a more limited memory ceiling (128GB vs. 512GB on max Mac Studio), but is meaningfully faster on inference for the same memory footprint. It runs CUDA, which means broader software compatibility (vLLM, the entire NVIDIA software ecosystem). For teams that already work in CUDA, this is the obvious choice. For teams in the Apple ecosystem, Mac Studio remains attractive.

Desktop server-class: Dell Pro Max with NVIDIA GB300

Dell shipped the Pro Max with NVIDIA GB300 (https://www.dell.com/en-us/dt/corporate/newsroom/announcements/detailpage.press-releases~usa~2026~03~dell-technologies-first-to-ship-nvidia-gb300-desktop-for-autonomous-ai-agents-with-nvidia-openshell.htm) in March 2026 — the first OEM to deliver a GB300 Grace Blackwell Ultra desktop. Specifications:

- 748GB of coherent memory
- 20 petaFLOPS at FP4
- 16TB of NVMe storage

- Ships ready to run NVIDIA's NemoClaw / OpenShell stack for always-on local AI agents

This is roughly six times the memory and twenty times the throughput of the DGX Spark. It targets a different workload class: where the alternative would otherwise be a small server room. Pricing depends on configuration but typically lands in the $30,000-$60,000 range — substantial but a fraction of what a comparable rack of GPUs costs.

What it's good for: Multi-user serving. Running the largest open-weight models (DeepSeek V4) without compromise. Fine-tuning at scale on a single machine. Sandboxed always-on agent workloads via NemoClaw. Replacing the role of a small cluster.

When it's overkill: Single-developer workflows. Cases where a Mac Studio or DGX Spark would handle the load. The price premium over those is real; only invest at this tier when you genuinely need the memory or throughput.

Consumer GPUs as a multi-user serving option

For shops standing up multi-user serving rather than single-developer workflows, a different configuration makes sense: consumer-grade NVIDIA GPUs (RTX 4090 or 5090 class) running quantized 70B models via vLLM (https://github.com/vllm-project/vllm). A two-GPU box with 80GB total VRAM (2× RTX 5090) can serve a quantized 70B model to a small team comfortably for around $7,000-$10,000.

Quantization — the technique that compresses model weights into smaller bit-widths — used to be a painful tradeoff with noticeable quality loss. Recent work on 4-bit quantization has narrowed that penalty to within the noise on most tasks. AWQ, GPTQ, and the newer FP4 formats are all production-ready.

Rough hardware tier guide

NEED	RECOMMENDED SETUP	APPROX TOTAL
Solo developer / curious individual	Mac Studio M3 Ultra (mid config: 192GB)	$5,500
Solo operator / small business with on-prem need	DGX Spark	$4,700
Small team (3-10 users), CUDA-friendly	RTX 5090 ×2 + workstation case + vLLM	$9,500
Mid-size team or large-context workload	Mac Studio M3 Ultra max (512GB) or DGX Spark + Mac Studio combo	$11,000-$15,000

NEED	RECOMMENDED SETUP	APPROX TOTAL
Multi-user production, large open-weight models	Dell Pro Max GB300	$30,000-$60,000
At-scale serving	Multiple GB300s or rack with H100/H200	$100,000+

05

The software stack

Hardware is half the picture. The software you run on it determines whether the deployment actually works for your team. Four tools you should know.

Ollama

Ollama (https://ollama.com/) is the easiest way to run open-weight models locally. It installs in one command, downloads models with `ollama pull`, and serves an OpenAI-compatible API. For solo developers, prototypes, and most internal tools, Ollama is the right starting point.

Strengths: Trivial setup. Wide model selection. OpenAI-compatible API means existing code largely works unchanged. Great default for desktop deployments.

Limitations: Single-user oriented. Not ideal for high-throughput multi-user serving (vLLM is better there). Limited control over batching, queuing, and observability.

Use when: You're a single developer, a small team running occasional inference, or you need a quick proof-of-concept before deciding on production architecture.

Apple's MLX

MLX (https://github.com/ml-explore/mlx) is Apple's machine-learning framework for Apple Silicon — the equivalent role to NVIDIA's CUDA stack on the Apple side. It's the foundation that Mac Studios and Mac-based ML workflows run on.

Strengths: First-class Apple Silicon performance (vs. running through compatibility layers). Active development. Open source. Strong tooling for fine-tuning on Apple hardware.

Limitations: Apple-only. Smaller community than CUDA ecosystem. Some advanced inference optimizations available in vLLM aren't yet in MLX.

Use when: You're committed to Apple Silicon and want maximum performance on that platform.

vLLM

vLLM (https://github.com/vllm-project/vllm) is the production-grade inference engine for NVIDIA hardware. It handles batching, KV-cache management, continuous batching, and tensor parallelism across multiple GPUs. For multi-user production serving on NVIDIA hardware, vLLM is the default.

Strengths: State-of-the-art throughput on NVIDIA. Production features: queue management, observability, OpenAI-compatible API. Active development.

Limitations: NVIDIA-only (CUDA required). Operational complexity higher than Ollama. Not the right tool for desktop deployments.

Use when: You're serving multiple users in production on NVIDIA hardware. You need throughput optimization. You want the most active production-inference ecosystem.

llama.cpp

llama.cpp (https://github.com/ggml-org/llama.cpp) is the lightweight, C++-based inference runtime that started the open-weight model deployment story in 2023 and remains the de facto choice for embedded, edge, and resource-constrained deployments.

Strengths: Tiny footprint. Runs on almost any hardware including ARM, x86, with or without GPU acceleration. Supports many model formats and quantization schemes. Foundation that many other tools (including Ollama) build on.

Limitations: Lower-level than Ollama or vLLM. More configuration required. Less well-suited for high-throughput multi-user serving.

Use when: You're deploying to edge devices, embedded systems, or resource-constrained environments. Or when you need fine-grained control over inference behavior.

Recommended pairings

HARDWARE	SOFTWARE STACK	BEST FOR
Mac Studio M3/M5 Ultra	MLX + Ollama for development; MLX server for production	Apple-ecosystem teams, single-user to small-team workloads
DGX Spark	Ollama for development; vLLM or NVIDIA NemoClaw for production	NVIDIA-aligned single-developer workflows
RTX 4090/5090 workstation	vLLM	Multi-user team serving
Dell Pro Max GB300	NemoClaw + OpenShell for agent workloads; vLLM for raw inference	Large-scale local serving, always-on local agents
Edge / embedded	llama.cpp	Constrained deployments

06

The workload decision matrix

This is the most important section of the guide. Before evaluating hardware or models, evaluate whether your workload is one that benefits from local deployment at all. We use this matrix when scoping engagements; it's blunt but accurate.

Yes, local is the right fit

Workloads where local model weights are now competitive with closed APIs and the data-control upside is meaningful:

- **Long-document summarization on a defined corpus.** DeepSeek V3.2/V4 handle ultra-long context excellently. Local removes data-flow concerns for sensitive corpora.
- **Structured extraction from semi-structured inputs.** Contract clauses, invoice line items, lab notebook entries, regulated filings. Open-weight models handle these cleanly; local removes data exposure.
- **Internal Q&A over a known set of documents.** Knowledge bases, technical documentation, research archives. Vector retrieval + a 70B-class local model produces production-quality results.
- **Code completion in private repositories.** DeepSeek V4 and Qwen 3 235B are at or near closed-frontier on SWE-bench. Repository code never needs to leave your network.
- **First-pass classification on high-volume queues.** Customer support routing, ticket triage, content moderation. Local models handle the routine 80% at near-zero marginal cost; escalate edge cases to closed APIs or human review.
- **Steady-state high-volume workloads.** Anywhere the API bill at expected volume would exceed $50K/year — the math usually favors local hardware amortized over 24-36 months.

Maybe local — case by case

Workloads where local is viable but the answer depends on specifics:

- **Bounded agentic loops** with well-defined tools and short context. Open-weight models handle these reasonably; closed frontier handles them better. The choice depends on data sensitivity and how complex the agent is.
- **Fine-tuned domain Q&A** where you have enough labeled data to make the fine-tune worthwhile. If you have 5,000+ high-quality examples, fine-tuning a smaller open-weight model often beats a frontier API on the specific domain.
- **Cost-driven decisions** at moderate scale. If your API bill is in the $10K-$50K/year range, the math is borderline and depends on engineering capacity.

Probably not local — but a local agent might still be the right answer

Workloads where running open weights locally is still a downgrade, but a local agent calling out to hosted APIs may still make sense:

- **Frontier reasoning** that benefits from the latest closed-model capabilities. The most recent Claude, GPT, or Gemini release may be meaningfully better on multi-step reasoning than the best open-weight option.
- **Multi-hour goal-driven agents** running over the open web. The integration surface alone (browsers, web tools, search) is a reason to use a hosted product like Codex Desktop with Atlas.
- **Multimodal video at frontier quality.** Open-weight video generation lags closed providers significantly.
- **Fresh-data agents** where you need the latest model release as soon as it ships.

For these workloads, consider the *local agent + hosted model* architecture instead: an agent process that runs on your hardware (Claude Code, OpenAI Codex Desktop, an OpenCUA agent, a NemoClaw setup) but calls out to hosted APIs for the LLM intelligence. Same data-handling caution applies — anything that goes into the prompt goes to the hosted provider — but you keep control over agent behavior, file access, and tool integration.

How to use this matrix

01 List the AI workloads your team currently runs (or is planning to run).

02 Categorize each one against the three buckets above.

03 For each "yes" workload, sum the expected volume in tokens or calls per month. If the total at API pricing would be >$30K/year, local hardware is likely cost-effective. Below that, the math is closer; lean local only if data sovereignty is a binding constraint.

04 For each "maybe" workload, run a 4-week pilot on the closest open-weight model to see if quality holds.

05 For "probably not local" workloads, default to closed APIs unless you have a hard data-flow constraint.

This matrix changes. It's accurate as of mid-2026. The set of workloads in the "yes" column has grown every six months for the past two years and we expect this to continue. Re-evaluate at least annually.

07

Worked example — research lab evaluating local AI deployment

A fictional but representative scenario. Numbers are illustrative; the analysis is real.

The setup

A university-affiliated research lab. 38 graduate students, 4 postdocs, 1 PI. Institutional research focus on environmental epidemiology — they work with anonymized but legally restricted health and demographic data spanning multiple jurisdictions. The lab wants AI assistance for: (1) literature review and summarization of recent papers, (2) Q&A over their internal corpus of ~15,000 indexed papers and reports, (3) help with code (R, Python) on their analysis pipelines, (4) draft support for grant applications and manuscripts.

Constraints:

- Research data is bound by IRB protocols and cannot leave their on-premises network.
- Some collaborators are in jurisdictions (EU, GCC) where data residency requirements affect what tools they can use.
- IT support is one staff person at 30% time.
- Budget for AI infrastructure: capital expenditure of $25K, ongoing operating budget of $4K/year.

The decision

Three options on the table:

- **Option A** — Closed-API access via institutional license (ChatGPT Enterprise / Claude Enterprise): $42K/year for the team. Data residency partially addressable but not fully. IRB review process estimated at 3-6 months.
- **Option B** — Single Mac Studio M3 Ultra (mid config, 192GB) running Llama 4 Maverick + Qwen 3 235B (alternating per workload). Hardware: $5,599. No recurring software cost. Engineering time to deploy: estimated 4-6 weeks of IT staff + 2 grad students.
- **Option C** — DGX Spark running a similar model setup. Hardware: $4,699. Same software stack (vLLM-based serving). Engineering time: similar.

Analysis

Step 1: Categorize workloads against the decision matrix.

WORKLOAD	BUCKET	WHY
Literature summarization	Yes-local	Long-context summarization, sensitive corpus, large volume
Internal Q&A	Yes-local	Defined corpus, sensitive content, retrieval-friendly
Code help (R, Python)	Maybe-local	Open-weight strong on code; not sensitive data
Manuscript / grant drafting	Yes-local	Drafts include unpublished research findings

All four workloads land in yes-local or maybe-local. None are in the probably-not-local bucket. Local deployment is structurally appropriate.

Step 2: Score the three options.

- **Option A (closed API)**: Best quality. Worst on data sovereignty (the binding constraint). Highest ongoing cost. Failed on IRB review for two of the four workloads.
- **Option B (Mac Studio)**: Strong on Apple Silicon. 192GB allows running 70B-class models comfortably. MLX framework appropriate. Single-team workload fits well within capacity.
- **Option C (DGX Spark)**: Slightly more raw inference performance per dollar. CUDA ecosystem broader. But the lab's software stack already standardized on Python + R; no advantage from CUDA. Memory ceiling lower than Mac Studio max.

Step 3: Pilot the leading option.

The lab pilots Option B for 6 weeks. They:

- Set up Ollama on the Mac Studio for development.
- Build a held-out test set: 30 questions answerable from their corpus, 15 paper-summarization tasks, 20 code-help cases.
- Run Llama 4 Maverick, Qwen 3 235B, and DeepSeek V3.2 against the test set.
- Score for accuracy, faithfulness to source, and code correctness.

Results: Llama 4 Maverick wins on general literature summarization. DeepSeek V3.2 wins on long-document Q&A (consistent with its long-context strength). Qwen 3 235B wins on code help. They decide to run all three — switching between them per query type via a simple router.

Step 4: Production deployment.

- Mac Studio M3 Ultra with 192GB RAM: $5,599
- Storage upgrade: $400
- UPS, networking gear: $400
- Total hardware: $6,399
- IT staff time to deploy and configure: ~4 weeks (within scope of existing role)

The lab's vector search infrastructure (built on PostgreSQL + pgvector) feeds the local LLM. Total user-facing latency for typical queries: 4-8 seconds. Acceptable for the use case.

Step 5: 24-month review.

At 24 months: hardware costs $6,399 (one-time). Operating costs: ~$1,200 (electricity, occasional IT time for model upgrades). Total: $7,600 over 24 months.

Compared to Option A: $84K over 24 months ($42K × 2 years), without solving the IRB / sovereignty constraints.

The lab saved $76K and got a deployment that actually meets their compliance posture. The IT staff person reports 2-4 hours/month of ongoing maintenance, well within capacity.

What you should take from this

Three lessons:

01 **Categorize workloads before evaluating tools.** They didn't pick hardware first; they categorized workloads first. The right hardware fell out of the decision matrix.

02 **The cost case for local strengthens at moderate scale.** $42K/year of closed-API spend turned into $6,400 of hardware over 24 months. The break-even is somewhere around the 6-month mark.

03 **Pilot before committing.** The 6-week pilot let them test three models on real data and discover that no single model was best at everything. The router pattern they adopted came from the pilot, not the planning phase.

08

Operational reality — what month four looks like

The local-AI buying decision is rarely about the hardware itself. It's about whether your team can operate it. Here's what month four typically looks like, by tier.

Single workstation (Mac Studio or DGX Spark)

Maintenance burden: Low — typically 1-3 hours per month.

- OS updates and inference framework upgrades (~30 minutes monthly).
- Model swap when a meaningfully better release lands (~1 hour per swap; most teams do this every 2-3 months in 2026).
- Storage management as you accumulate models and KV cache (~30 minutes monthly to clean up).

Common surprises:

- Disk fills up faster than expected — open-weight models are 50-200GB each, and you may want multiple loaded.
- Inference performance varies with thermal management, especially in Mac Studios under sustained load.
- Some open-weight models ship with subtle behavior differences from closed APIs that surface in production (different chat templates, different stop tokens, etc.). First couple of weeks of debugging.

What "fails" looks like: If you don't have someone — even a single staff person at 25% time — owning this, the deployment usually goes idle within 4-6 months because nobody updates models or fixes the small issues that accumulate. The hardware sits there. The team falls back to closed APIs. The "local AI strategy" quietly fails.

Multi-GPU consumer setup (RTX 4090/5090 + vLLM)

Maintenance burden: Moderate — typically 4-8 hours per month.

- vLLM updates and model migrations.
- Throughput tuning as load patterns evolve.
- KV cache and batching configuration adjustments.
- GPU driver / CUDA upgrades occasionally cause issues.

Common surprises:

- Power draw is real. Two 5090s under sustained load require attention to electrical capacity and cooling.
- Inference quality at production traffic loads can differ from pilot loads (saturated batching changes latency profile).
- The team often picks the wrong quantization at first; getting from "models load" to "models load and run at expected quality" takes a few iterations.

What "fails" looks like: Without a dedicated devops / ML-engineering person, multi-GPU setups often run sub-optimally — the throughput is much lower than the hardware is capable of. This isn't a binary failure; it's a slow degradation where users gradually prefer the closed API because it feels faster.

Server-class (Dell Pro Max GB300, multi-node setups)

Maintenance burden: Substantial — typically 0.25-0.5 FTE.

- Real operations work: monitoring, alerting, incident response, capacity planning.
- Software stack: NemoClaw / OpenShell / vLLM / cluster management.
- User support: onboarding, access management, debugging issues.

What "fails" looks like: Underused hardware. The Dell Pro Max GB300 has the capacity to serve hundreds of users; if your actual user base is 10 developers, you bought too much hardware. Right-sizing is critical — most teams over-buy at this tier.

Model migration — the recurring operational cost most teams underestimate

Every 3-6 months, a meaningfully better open-weight model gets released. You'll want to migrate. What that involves:

01 Run your held-out test set against the new model. Compare scores against the current production model.

02 If the new model wins, update the chat template, system prompts, stop tokens, and any model-specific quirks.

03 Re-run the test set on the production-equivalent setup.

04 Roll out gradually — 5%, 25%, 100% of traffic — watching the production observability dashboard for regressions.

05 Hold the old model loaded for 1-2 weeks as a fallback before fully decommissioning.

This whole process takes 1-3 days of engineering time per migration. If you don't have an eval suite (see Guide № 03 — *Evals before code*), step 1 is impossible and you're flying blind.

When to upgrade hardware

Two triggers:

01 **Your largest model can't fit in memory.** A new open-weight model lands and you can't load it. Time for more RAM or a bigger machine.

02 **Latency at peak load is unacceptable.** When users are abandoning queries because they're too slow, you've outgrown the current hardware.

Don't upgrade based on "the new chip is X% faster." Upgrade based on observed user-facing problems. The Mac Studio M5 Ultra will be a meaningful jump from M3 Ultra, but M3 Ultra is still excellent for most workloads — don't churn hardware unnecessarily.

09

Who actually benefits

Local AI is rarely the right answer because it's cheaper. It's the right answer because of control. Five distinct buyer profiles make the decision differently.

Sensitive-data environments. Research labs working with subject data. Healthcare providers with patient records. Government with classified material. Legal teams with attorney-client privileged content. The regulatory line is often not negotiable. Local is the only way to get useful AI inside it.

Sovereignty-driven procurement. Some regions and industries treat data sovereignty as a default rather than an exception. The Gulf is one example, where local-first AI is increasingly part of the procurement conversation; large parts of European public-sector and defense procurement are similar. The question for these buyers is rarely "is local as good as cloud?" — it is "what works given that cloud is not an option?"

Offline-by-design contexts. Drone operations in regions without reliable connectivity. Autonomous mobility with onboard inference. Vessels, rigs, vehicles, and field deployments without high-bandwidth uplinks. The workload shape simply requires local.

Solo operators and small businesses. A single owner running their own books, a small consultancy automating internal workflows, an independent professional building an AI assistant against their own files. For this group, cost is often the deciding factor — running a 70B-class model on a Mac Studio amortizes over months, while equivalent API spend at the same query volume can be substantial. Privacy is a secondary motivator.

Personal AI builders. People who want an AI assistant on their own machine — running while offline, learning from their own files, executing tools without going through a third party. The hardware to run a usable system at home is now broadly available; the software stack improves each month. For this group, local AI is less about cost or compliance and more about owning the workflow.

If none of these descriptions fit your situation, local AI may not be the right answer for you. A frontier API will serve you better. That's not a value judgment; it's a workload-fit observation.

APPENDIX A

Hardware quick-reference

For the procurement conversation. Verified as of mid-2026; prices subject to change.

HARDWARE	MEMORY	FLOPS / TOPS	APPROX PRICE	BEST FOR
Mac Studio M3 Ultra (base 96GB)	96GB unified	—	$3,999	Solo developer, 30B-class models
Mac Studio M3 Ultra (192GB)	192GB unified	—	$5,599	Solo dev / small team, 70B comfortable
Mac Studio M3 Ultra (max 512GB)	512GB unified	—	$10,799	Heavy local workloads, full open-weight access
NVIDIA DGX Spark	128GB unified	1 PFLOP FP4 / 1000 TOPS	$4,699	NVIDIA-aligned single dev, agent workloads
RTX 5090 ×2 + workstation	64GB VRAM	~350 TOPS each	$9,500	Small-team multi-user CUDA serving
Dell Pro Max with GB300	748GB coherent	20 PFLOPS FP4	$30K-$60K	Multi-user, largest open-weight, server-class

Coming later 2026:

— **Mac Studio M5 Ultra** — expected mid/late 2026, up to 256GB unified memory, ~1100 GB/s bandwidth, dedicated GPU matrix-multiply units. Expect significant inference speed improvements.

APPENDIX B

Open-weight model recommendations by workload

Verified as of mid-2026. Re-verify quarterly; the open-weight landscape moves quickly.

WORKLOAD	FIRST CHOICE	SECOND CHOICE	NOTES
General-purpose mixed	Llama 4 Maverick	DeepSeek V4	Best balance of quality and footprint
Long-document summarization	DeepSeek V4	Llama 4 Maverick	V4 architecture optimized for ultra-long context
Reasoning-heavy	Qwen 3 235B	DeepSeek V4	Qwen leads at this tier
Code completion	DeepSeek V4	Qwen 3 235B	Both top open-weight on SWE-bench
Memory-constrained (single workstation)	Qwen 3-35B-A3B	Llama 4 Maverick (MoE-routed)	Active params smaller than total
European deployment (compliance)	Mistral Large 2 / 3	Llama 4 Maverick	Mistral's compliance posture matters in EU
Edge / embedded	Gemma 4 (smaller variants)	Mistral Small 3	Smaller models with reasonable quality

APPENDIX C

About Oasium AI

Oasium AI is an applied AI consultancy. Two PhDs (UC Berkeley and UCLA), based across San Francisco and Dubai, who use AI in their own daily work — and help organizations do the same. Three service lines: **Educate** (workshops and training), **Strategize** (AI feasibility and roadmaps), and **Implement** (production AI workflows, agent systems, **on-prem deployments** of the kind this guide describes).

We wrote this guide because the local-vs-cloud conversation is one we have weekly with research labs, healthcare clients, and sovereignty-focused organizations across the GCC, EU, and US. Most published guides on this topic are written by hardware vendors trying to sell hardware or cloud providers trying to keep your data flowing through their APIs. We don't sell hardware. We don't take partner commissions. We just do the work.

If you're sitting in front of a local-AI deployment decision and want a structured second opinion — including honest input on whether local is the right answer for your specific case — we offer free 30-minute discovery calls. No slides, no script, just a conversation.

Book a discovery call → (https://oasium.ai/contact#book-a-call)

Or write us at hello@oasium.ai.

If this guide was useful and you'd like more like it, the rest of our writing lives at oasium.ai/writing (https://oasium.ai/writing). We post when something's worth saying — usually a few times a quarter.

— Ziad and Jonah Co-founders, Oasium AI

Oasium AI

COLOPHON

About this guide

An opinionated buyer's guide for teams considering on-prem AI deployments in 2026. Hardware tiers (Mac Studio, DGX Spark, Dell GB300), open-weight model picks per workload, the software stack, a workload decision matrix, a worked example, and the operational reality of month four.

AUTHORS

Ziad Yassine and **Jonah Lipsitt**

Co-founders, Oasium AI

San Francisco + Dubai

CONTACT

oasium.ai

hello@oasium.ai

For consulting work, the workshop, or to point out something we got wrong.

EDITION

Guide № 02 · May 2026

The latest version is always at oasium.ai/guides.