Local Three-Tier Agent Workstation

Matching agent framework and open-weight model architecture to the workload, on a single 32 GB GPU

Design & operator setup – architecture and fit measured; per-tier throughput and swap-cost stated as falsifiable criteria pending own benchmarks

Abstract

A single 32 GB GPU cannot hold the models needed for three very different agent workloads at once, and it should not try to. This document describes an operator setup that instead partitions the work into three tiers – an interactive cognitive cockpit, an agile software-engineering environment, and an autonomous long-horizon factory – and pairs each tier with the agent framework and open-weight model whose architecture fits the workload's cost model: test-time compute for reasoning, sparse activation for coding throughput, a small dense footprint for long-context background work. Because the smallest viable pair of models already exceeds the 32 GB ceiling, the design is built on strict single-residency: exactly one model is resident, and a tier switch swaps it. The paper situates this against the local-serving, model-routing, and cold-start literature, states the fit envelope that forces the design and the swap cost it incurs, turns its central claims into falsifiable acceptance criteria with instruments and thresholds, and treats safety as a cross-cutting requirement that is most stringent precisely where the design is most capable – the unattended, high-privilege Tier 3, whose chosen tooling has a documented, severe security record.

Keywords: local agents · open-weight models · Mixture-of-Experts · model routing · VRAM budget · single-residency · cold-start latency · agent security · sovereign compute

§01

The problem

The temptation with a capable local card is to run one large model for everything. That is the wrong optimisation. Three agent workloads – reasoning-heavy research and drafting, repository-scale software engineering, and unattended long-horizon automation – reward three different architectural properties, and paying for all of them at once wastes both VRAM and latency. The constraint that shapes everything is memory: at Q4, the relevant open-weight models occupy 15–24 GB each, so no two fit together on 32 GB. The design therefore does not co-locate models; it time-multiplexes a single resident model across three tiers, and spends its engineering budget on choosing the right model per tier and on making the swap between them cheap and safe.

§02

Background and related work

Three strands of the systems and modelling literature define the design space this setup operates in. First, **efficient local serving and the memory-bandwidth wall.** On a single consumer card, autoregressive decode is bound by memory bandwidth, not compute – each generated token reads the active weights once through the memory bus – and paged-attention serving [6] is what makes long-context inference tractable at all. This is also why sparse Mixture-of-Experts models, which read only their active experts per token [13], deliver dense-model competence at a fraction of the per-token bandwidth cost, and why they anchor the coding and allrounder tiers here.

Second, **routing the right model to the task.** Work such as RouteLLM frames model selection as a routing problem – send each query to the cheapest model that can answer it, escalating only when needed [15]. This setup applies the same principle under a hard hardware constraint: instead of routing across a fleet, it routes across time

on one card, matching each workload to the model architecture whose cost it can pay, and accepting a swap where a datacentre would simply hold both models resident.

Third, and most specific to the design's cost, **model-loading cold-start.** The swap between tiers is exactly the cold-start problem studied in serverless LLM serving, where loading a multi-gigabyte checkpoint into GPU memory dominates first-token latency and can run from tens of seconds to minutes; systems like ServerlessLLM attack it with multi-tier checkpoint caches and locality-aware loading [14], and a parallel line of MoE-serving work reduces the resident footprint further through expert offloading and caching [16]. This setup does not solve cold-start in general; it budgets it, on a single card, and states the resulting latency honestly (§7). Against this landscape, the contribution is not a new serving system but a disciplined operator composition: a workload-to-architecture mapping plus a single-residency swap policy that a solo operator can actually run and reason about on commodity hardware.

§03

The three tiers

Each tier pairs a workload with the agent framework and the open-weight model class whose architecture fits it (Figure 1). Only one model is resident at a time; a tier switch swaps it.

Tier	Workload	Agent framework	Model class
1 · Cognitive cockpit	Everyday & research; reasoning-heavy writing	Hermes (Nous Research) – memory, autonomous skills, isolated subagents [1]	Dense reasoning (DeepSeek-R1 32B) or fluent generalist (Gemma4-31B / Qwen3.6-35B-A3B)
2 · Agile dev env	Repo-scale software engineering	OpenCode – terminal-native, reads the working dir	Coding MoE (qwen3-coder:30b)
3 · Autonomous factory	Unattended long-horizon jobs	OpenClaw – persistent, checkpoint/resume, high-privilege	Efficient specialist (Devstral Small 2, 24B) or allrounder (Qwen3.6-35B-A3B)

Three tiers – match agent and model architecture to the workload

One 32 GB card, three distinct workloads; each tier pairs the agent framework and the model whose architecture fits.

§04

Architecture-to-workload: why these model classes

The tier assignments are not arbitrary; each rests on a property of the model architecture that the workload rewards, and on refusing to pay for a property the workload does not use.

- **Reasoning (Tier 1 research) → a dense RL-trained reasoning model.** DeepSeek-R1 32B spends test-time compute on a long autoregressive chain-of-thought, self-correcting before answering; it trades tokens – and therefore latency – for correctness on hard problems [3]. That trade is worth making for research, where one well-reasoned answer beats a fast shallow one, but it means the effective time-to-answer is longer than raw tokens/s implies. That is a Tier-1 design acceptance, not a defect.
- **Drafting (Tier 1 writing) → a fluent generalist.** Gemma4-31B or Qwen3.6-35B-A3B, where reasoning depth matters less than throughput and prose quality.
- **Software engineering (Tier 2) → a coding MoE.** qwen3-coder's sparse routing activates ~3.3 B of 30.5 B parameters per token, giving heavy-model competence at light-model latency: because generation is memory-bandwidth-bound and an MoE reads only its active weights per token [13], it sustains 2–4× the token rate of a dense model of comparable total size on the same card (§6) [4]. Its long native context holds a repository working set so multi-file review does not thrash the KV cache.
- **Long-horizon (Tier 3) → an efficient specialist or allrounder.** Devstral Small 2 or Qwen3.6-35B-A3B, where fault-tolerant orchestration and long context matter more than latency – and where Devstral's small 24 B footprint leaves the most KV headroom for extended jobs (§6) [2].

The unifying rule: pick the architecture whose cost model the workload can pay – test-time compute for reasoning, sparse activation for coding throughput, a small dense footprint for long-context background work – and never spend VRAM or latency on a property the tier does not use.

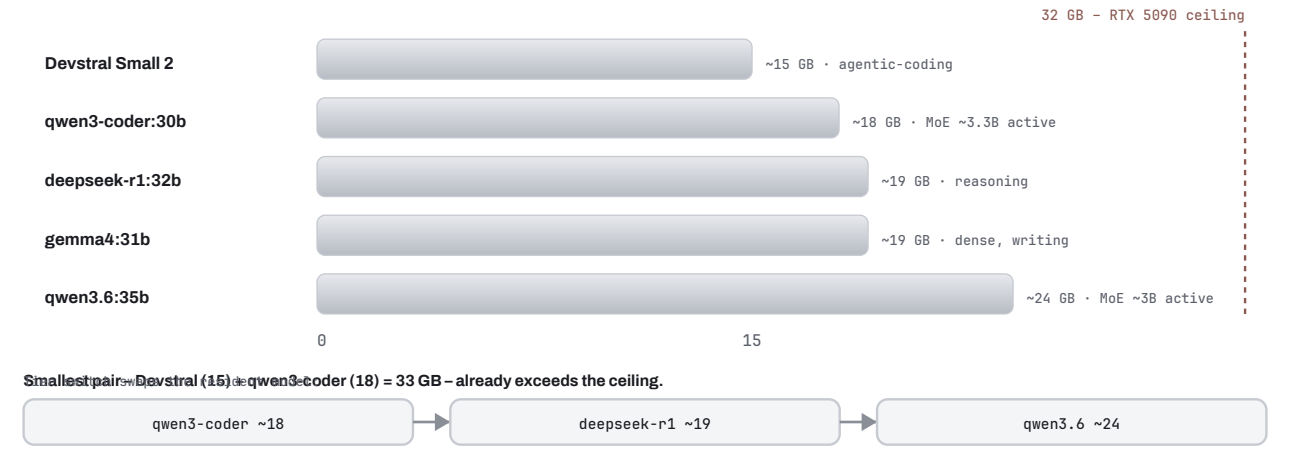
§05

Fitting 32 GB: one model at a time

Any single model fits with headroom; no two fit together – which is why the setup time-multiplexes rather than co-loads (Figure 2). At Q4 the resident weights run 15–24 GB, so the smallest viable pair – Devstral (~15 GB) plus qwen3-coder (~18 GB) = ~33 GB – already exceeds the 32 GB ceiling. Single residency is therefore not a limitation worked around but the design itself: Ollama enforces it with `OLLAMA_NUM_PARALLEL=1` and `OLLAMA_MAX_LOADED_MODELS=1`, queues incoming work, and unloads the resident model before loading the next on a tier switch.

Fitting 32 GB – one model at a time

Any single model fits with headroom; no two fit together – which is why the setup time-multiplexes.



Performance and context envelope

Two numbers decide whether a tier is usable in practice: how fast it generates, and how much context it can hold – and on a single 32 GB card the second constrains the design far more than model size alone suggests (Figure 3). The table gives indicative published figures for these model classes on the RTX 5090 (Q4, single stream); they are external benchmarks for the architecture, not the author's own deployment measurements, which are the subject of the acceptance criteria in §8.

Model	Weights	Generation (tok/s)	Prefill (tok/s)	Practical context in 32 GB
Devstral Small 2 (24B)	~15 GB	~87	~14,500	very long – ~17 GB KV budget (256K feasible)
qwen3-coder:30b (MoE)	~18 GB	~234 → ~110 (4K→32K)	~3,000+	~128–147K measured within VRAM
deepseek-r1:32b (dense)	~19 GB	~52–94	~1,450 (@32K)	~47K – dense KV fills the budget fast
gemma4:31b (dense)	~19 GB	~61	~1,170	moderate
qwen3.6:35b (MoE)	~24 GB	~110 → ~234	high	short – only ~8 GB KV budget

Two design consequences follow. First, **generation is memory-bandwidth-bound**: the 5090's ~1.79 TB/s means an MoE that reads only ~3 B active weights per token sustains 2–4× the token rate of a dense model of similar total size – which is exactly why the coding and allrounder tiers use MoE, and why the dense reasoning model is the slowest per token (and slower still in effective time-to-answer, because it emits a long chain of thought before the answer). Second, **the context ceiling is a per-tier budget, not a constant**: weights consume the base, and the remaining 32 GB – weights is the KV-cache budget. The 15 GB Devstral leaves ~17 GB for context (a genuine 256K working set), whereas the 24 GB qwen3.6 leaves only ~8 GB and must run a much shorter context. The earlier "fits with headroom" is therefore refined: every model fits, but the usable context per tier differs sharply, so the operator budgets context per tier rather than assuming a uniform ceiling. A q8_0 KV cache roughly halves this pressure, as established in the evaluation-pipeline work [12].

Performance & context budget on the 5090

Weights set the floor; the remainder of 32 GB is the KV-cache / context budget.

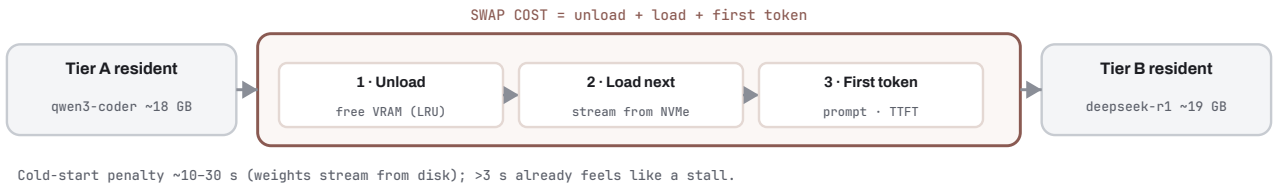
§07

Swap discipline and its cost

Single-residency buys "it fits" at the price of a cold swap on every tier switch, and honesty requires naming that cost rather than eliding it (Figure 4). A tier switch is three steps: unload the resident model (LRU eviction frees VRAM), load the next model (weights stream from NVMe into VRAM), and process the first prompt (TTFT). At the model sizes here (15–24 GB), the reload dominates and the cold-start penalty is on the order of ten to thirty seconds – consistent with the serverless-serving literature, where checkpoint loading, not compute, dominates first-token latency [14] – and since users perceive waits beyond a few seconds as stalls [10], this latency is the real ergonomic tax of the design.

The swap lifecycle, and its cost

Single-residency buys “it fits” at the price of a cold swap on every tier switch.



Cold-start penalty ~10–30 s (weights stream from disk); >3 s already feels like a stall.

The tax is manageable, and the mitigations are part of the design rather than afterthoughts. The most-used model can be kept warm via `OLLAMA_KEEP_ALIVE` (or `keep_alive: -1`), and the next tier's model can be pre-warmed before an anticipated switch – the single-card analogue of the checkpoint-cache and locality strategies used by serverless systems [14]. Two honest constraints remain. First, there is no true cross-tier parallelism under strict single-residency: an unattended Tier-3 job holds the card, so interactive Tier-1 work waits unless the operator schedules around it – the single-GPU analogue of the co-residency limit. Second, CUDA memory fragmentation accumulates over repeated load/unload cycles, so free contiguous VRAM (not just free total) must be monitored and the server restarted periodically. The alternative design – pinning one small always-on model beside a swapped large one – trades part of the fit budget for lower switch latency, and is a reasonable variant for operators whose Tier-1 use is near-constant.

§08

Acceptance criteria (evaluation plan)

Because this is a design & operator setup rather than a measured benchmark study, its central claims are stated here as falsifiable acceptance criteria, each with an instrument and a pass/fail threshold, to be evaluated on the running setup – the same measured/predicted discipline the companion papers use. Thresholds marked [proposed] are defaults offered for confirmation. The workstream hypothesis (W3) is: single-residency time-multiplexing serves three model tiers in 32 GB without VRAM collision, at a bounded and manageable swap cost. It decomposes into five criteria.

- **A1 – No-collision single residency.** Metric: peak VRAM and host-RAM spill under each tier's representative workload at its budgeted context. Instrument: GPU memory telemetry across a scripted run of all three tiers. Pass: peak VRAM below the 32 GB ceiling with headroom and zero spill to system RAM on every tier. Falsification: any spill or OOM under a tier's stated context budget. (predicted; the single-model fit is measured, whole-workload confirmation pending)
- **A2 – Bounded swap cost.** Metric: cold-swap latency (unload + load + first token) and warm-tier (keep-alive) switch latency, over N repeated switches. Instrument: timed tier switches from NVMe. Pass [proposed]: median cold swap ≤ 30 s at these model sizes; a kept-warm tier switches in ≤ 3 s. Falsification: swap latency beyond the budget, making interactive tier-switching unusable. (predicted)

- **A3 – Per-tier throughput floor.** Metric: sustained single-stream generation rate per tier at its budgeted context. Instrument: a fixed generation benchmark under `NUM_PARALLEL=1`. Pass [proposed]: the MoE tiers ≥ 100 tok/s and the dense reasoning tier ≥ 40 tok/s (above reading speed, accepting the reasoning latency trade). Falsification: a tier below its usability floor. (predicted; §6 figures are external, own measurement pending)
- **A4 – Per-tier context budget realised.** Metric: maximum usable context per tier without OOM under a q8_0 KV cache. Instrument: a context-length sweep to the OOM boundary. Pass: each tier reaches its budgeted context (e.g. ~256K for the 15 GB Devstral, a short context for the 24 GB qwen3.6) with no spill. Falsification: OOM below the budgeted context. (predicted)
- **A5 – Isolation efficacy (Tier 3).** Claim: the sandbox contains a hostile autonomous agent. Instrument: red-team the Tier-3 substrate with the known OpenClaw attack classes – indirect prompt injection driving an action, a malicious imported skill, and sandbox-escape attempts – under the hardened container (§9). Pass: zero network egress, zero sandbox escape, every sensitive action gated. Falsification: any containment breach. (predicted)

All five are runnable on the same single-GPU, air-gapped setup plus the evaluation harness already in the program; A5 in particular reuses the W1 hardened sandbox directly [12].

§09

Safety across tiers

The highest-risk tier is Tier 3: an unattended, long-horizon, high-privilege autonomous agent. This is not hypothetical for the chosen tooling. OpenClaw's rapid rise came with a severe, well-documented security record: a critical remote-code-execution flaw via an unvalidated local WebSocket (CVE-2026-25253) that let any visited website reach the local agent, a chained set of sandbox-escape and privilege-escalation vulnerabilities ("Claw Chain", CVE-2026-44112/44113/44115/44118, the allowlist-bypass rated CVSS 8.8), and a marketplace supply-chain problem – an independent audit of its skill marketplace found 341 malicious entries among 2,857 skills, designed to steal credentials, open reverse shells, and hijack agents [7, 8]. A default posture of broad privilege and minimal sandboxing maps to a concrete threat set: indirect prompt injection, skill supply-chain contamination, memory poisoning, and intent drift [8, 9]. Safety is therefore treated here as a cross-cutting requirement, applied most stringently where autonomy and privilege are highest, and verified – not assumed – by criterion A5.

- **Untrusted inbound data.** Email, web content, and any tool output are treated as untrusted across all tiers – the standard mitigation against indirect prompt injection, where the attack rides in on data the agent reads rather than on the operator's instruction [9].
- **Confirmation gates.** Sensitive or irreversible actions require explicit `[Y/n]` confirmation; on the unattended Tier 3, the equivalent is an allowlist of permitted actions plus checkpoint-based rollback if behaviour drifts from the task [7].
- **Execution isolation.** All model-driven execution runs under container isolation with dropped Linux capabilities, non-root, and no ambient network – the same hardened substrate established in the evaluation pipeline (no network, non-root, read-only filesystem, seccomp) [12]. For Tier 3 this is not optional hardening but the precondition for running the agent at all; a vendor enterprise layer (NVIDIA NemoClaw) exists precisely because the base tool needs sandbox orchestration added on top [11].
- **Memory and supply-chain integrity.** Because persistent memory and autonomously created/shared skills are attack surfaces (memory poisoning, skill supply-chain contamination), memory writes and imported skills are validated, and checkpoints provide a trusted state to roll back to [7, 8].

The honest position: the three-tier design is safe to run only to the extent that the isolation layer is real; the convenience of an autonomous background agent is exactly proportional to the discipline of the sandbox around it.

§10

Applied scenario (STAR)

A solo operator running three agent workloads on one card, sovereignly.

- **Situation.** An independent ML operator needs an interactive research-and-writing assistant, a repository-scale coding agent, and an unattended overnight automation agent – but has one 32 GB card, a data-sovereignty constraint (nothing leaves the machine), and no budget for a multi-GPU rig or per-workload cloud subscriptions.
- **Task.** Run all three workloads on the single card with each getting a model whose architecture actually fits its cost, without VRAM collisions, with predictable switch latency, and with the unattended agent safely contained.
- **Action.** Configure Ollama for strict single residency; assign DeepSeek-R1 / Gemma4 to Tier 1, qwen3-coder to Tier 2, Devstral to Tier 3; keep the most-used tier warm via `OLLAMA_KEEP_ALIVE`; budget context per tier under a q8_0 KV cache; and run every model-driven action – most stringently the Tier-3 agent – inside the hardened, network-less sandbox.
- **Result.** Three capable agent workloads served from one commodity card, each matched to its architecture, swapped rather than co-loaded, at a ten-to-thirty-second switch cost the operator can plan around – and an unattended automation tier whose safety rests on a verified isolation layer, not on trust in the agent framework. The setup is fully sovereign and re-derivable from the pinned configuration.

§11

Limitations and threats to validity

Four limitations bound the claims. First, **no true concurrency:** strict single-residency means an unattended Tier-3 job blocks interactive Tier-1 work; operators whose interactive use is near-constant should prefer the always-on-small-plus-swapped-large variant, at a cost in fit budget (§7). Second, **the performance figures in §6 are external, not measured here:** they are published benchmarks for these model classes, and the author's own single-stream numbers (A2–A4) are pending – the operational claims should be read as predicted until then. Third, **model availability and drift:** the specific open-weight models are a 2026 snapshot; the design is a mapping from workload to architecture class (dense-reasoning, coding-MoE, small-dense-long-context), which outlives any one checkpoint, but the exact footprints and rates will move as models are updated. Fourth, and most consequential, **safety is only as real as the sandbox:** the Tier-3 threat model is defended by container isolation whose efficacy is asserted until A5 is run against the concrete OpenClaw attack classes; a containment failure there would invalidate the "safe to run unattended" claim, which is why A5 is treated as load-bearing rather than a formality.

§12

Status & reproducibility

Architecture and fit are established; per-tier throughput, context budget, swap cost, and isolation efficacy are stated as the acceptance criteria of §8, to be measured next under the same pinned, air-gapped discipline as the companion papers. Reproducibility pins: `OLLAMA_NUM_PARALLEL=1`, `OLLAMA_MAX_LOADED_MODELS=1`, `OLLAMA_KV_CACHE_TYPE=q8_0`, `OLLAMA_FLASH_ATTENTION=1`, and `OLLAMA_KEEP_ALIVE` for the warm tier; per-tier model pinned by tag; tier switch = unload → load → first token. Built and operated under CTC AI Operations, on the same local-inference, single-residency, and hardened-sandbox discipline as the evaluation, code-evaluation, and assistant projects it sits beside; the isolation layer of §9 and the A5 test reuse the W1 sandbox directly.

References

[1] Nous Research. (2025). Hermes 4 – tool-use, persistent memory, and isolated subagents for local agents. Model card. [2] Mistral AI. (2025). Devstral Small 2 (24B) – agentic software-engineering model. Model card. [3] DeepSeek-AI. (2025). DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948. [4] Qwen Team. (2025). Qwen3-Coder – sparse Mixture-of-Experts coding model (~3.3B active / 30.5B total); long native context. Model card. [5] Qwen Team. (2026). Qwen3.6-35B-A3B – sparse MoE, hybrid gated-linear + gated attention. Model card. [6] Kwon, W., Li, Z., Zhuang, S., et al. (2023). Efficient Memory Management for LLM Serving with PagedAttention. SOSP. arXiv:2309.06180. [7] Cyera; The Hacker News; NVD. (2026). OpenClaw "Claw Chain" (CVE-2026-44112 / 44113 / 44115 / 44118; allowlist bypass CVSS 8.8) and CVE-2026-25253 (WebSocket RCE); ClawHub skill-marketplace audit (341 / 2,857 malicious skills). National Vulnerability Database and security disclosures, 2026. [8] Taming OpenClaw: Security Analysis and Mitigation of Autonomous LLM Agent Threats. (2026). arXiv:2603.11619. [9] UK National Cyber Security Centre. (2025). Prompt Injection Is Not SQL Injection (It May Be Worse). [10] Nielsen, J. (1993). Response Times: The 3 Important Limits. In Usability Engineering – a wait beyond ~1 s breaks flow; beyond ~10 s loses attention. [11] NVIDIA. (2026). NemoClaw – enterprise sandbox orchestration, privacy guardrails, and security hardening layered on OpenClaw. [12] Arenskrieger, M. E. (2026). Hybrid Evaluation Pipeline (CTC AI Operations, W1) – q8_0 KV cache and the hardened sandbox (no network, non-root, read-only, seccomp) reused for §9 and A5. [13] Fedus, W., Zoph, B., Shazeer, N. (2022). Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. JMLR. arXiv:2101.03961. [14] Fu, Y., Xue, L., Huang, Y., et al. (2024). ServerlessLLM: Low-Latency Serverless Inference for Large Language Models. OSDI. arXiv:2401.14351. [15] Ong, I., Almahairi, A., Wu, V., et al. (2024). RouteLLM: Learning to Route LLMs with Preference Data. arXiv:2406.18665. [16] Xue, L., Fu, Y., Lu, Z., et al. (2024). MoE-Infinity: Efficient MoE Inference on Personal Machines with Sparsity-Aware Expert Cache. arXiv:2401.14361.