✦ SOLOPRENEUR HARDWARE BIBLE · PRODUCTION BLUEPRINT

THE 8GB VRAM FIELD MANUAL

High-End AI Engineering on Real-World Consumer Hardware

Author: Stefan Luca · Architecture: Luca Automated · Web: https://lucaautomated.com · Nov 2026

✦ PHILOSOPHY & MANIFESTO

The tech industry preaches an expensive doctrine: that building production-grade AI systems, autonomous agent swarms, and photorealistic generative media requires $30,000 enterprise GPUs, rented H100 clusters, or $500/month recurring cloud API subscriptions.

We prove them wrong.

This manual contains the battle-tested blueprints of **Luca Automated**. Every system detailed here is designed, benchmarked, and executed locally on consumer-grade hardware:

- **Primary Inference & Diffusion Node:** NVIDIA RTX 3070 Ti (8GB GDDR6X VRAM / 6144 CUDA cores).
- **24/7 Sovereign Gateway & Dispatcher:** Refurbished Lenovo ThinkPad T440 ($50 / Intel Core i5 / 8GB RAM / 15W TDP).
- **Local Multi-Agent Network:** Ollama, llama.cpp, SQLite WAL queue, Python 3.10+, and PM2.

Total Cloud Subscription Budget: $0.00 / month.

MODULE 01 · THE 8GB VRAM MEMORY MATRIX & CUDA HARDENING

Running frontier models on an 8GB VRAM card without `CUDA out of memory` (OOM) errors requires ruthless memory budgeting.

The 8192 MB Memory Budget Breakdown

```
Total Physical VRAM:                 8,192 MB (8.00 GB)
NVIDIA Driver & Windows DWM Overhead: - 650 MB
Available Compute Budget:            7,542 MB
Safety Headroom Margin (5%):         - 377 MB
MAXIMUM ALLOCATABLE VRAM:            7,165 MB (~7.0 GB)
```

Quantization Hierarchy for 8GB VRAM

- **GGUF (llama.cpp / Ollama):** Best for hybrid CPU/GPU layer offloading.
  - **Q4_K_M (4-bit medium):** The gold standard for 7B-8B models (fits 100% in VRAM at 4.6 GB).
  - **Q3_K_M (3-bit medium):** The mathematical key to running 14B models on 8GB VRAM (model weights: 6.8 GB).

- **EXL2 (ExLlamaV2):** Highest throughput (tokens/sec) when 100% of layers fit strictly in VRAM. Use for 8B models at 4.25 bpw.
- **FP8 / BitsAndBytes NF4 v2:** Mandatory for modern diffusion transformers (Flux.1, SDXL).

MODULE 02 · DEEPSEEK-R1 & QWEN 2.5 14B ON 8GB VRAM

The 14-billion parameter class represents the frontier of open-weight reasoning. Standard 16-bit float requires 28 GB VRAM. Here is how to run it at **28 tokens/sec** on an RTX 3070 Ti.

The Empirical Benchmark Matrix

Model Variant	Quantization	GPU Layers (of 48)	VRAM Footprint	Throughput	Stability
DeepSeek-R1-Distill-14B	**Q3_K_M**	**40 / 48**	**7,050 MB**	**28.4 t/s**	**100% Rock Solid**
DeepSeek-R1-Distill-14B	Q4_K_M	28 / 48	7,120 MB (RAM overflow)	11.2 t/s	PCIe Bottleneck
Qwen 2.5 14B Instruct	**Q3_K_M**	**40 / 48**	**6,980 MB**	**29.1 t/s**	**100% Rock Solid**
Qwen 2.5 14B Instruct	Q4_K_M	28 / 48	7,110 MB (RAM overflow)	10.8 t/s	PCIe Bottleneck

> **Key Rule:** Avoid partial CPU offloading of large layer blocks. Q3_K_M allows 40 of 48 layers on GPU with a 4K context window, yielding 2.5x higher speed than Q4_K_M offloaded to system RAM.

Production llama.cpp Server Launch Command

```
./llama-server \
  -m ./models/DeepSeek-R1-Distill-Qwen-14B-Q3_K_M.gguf \
  --ngl 40 \
  -c 4096 \
  --flash-attn \
  --port 8080 \
  --host 127.0.0.1
```

MODULE 03 · FLUX.1 COMPILED DIFFUSION BLUEPRINT (COMFYUI)

Flux.1 (Dev and Schnell) has replaced SDXL as the apex open-source generative model, but its 12-billion parameter flow-transformer crashes standard 8GB setups immediately.

The Zero-OOM ComfyUI Architecture

1. **UNET / Diffusion Model:** Use `flux1-schnell-bnb-nf4-v2.safetensors` or `flux1-dev-Q4_K_S.gguf` (~6.2 GB).
2. **Text Encoder 1 (CLIP L):** Loaded strictly to VRAM (~250 MB).

3. **Text Encoder 2 (T5-XXL FP8):** Offloaded strictly to System RAM via ComfyUI Split CLIP Loader (`--lowvram`).
4. **VAE Decoder:** Run `Tiled VAE Decode` with tile size set to `512x512` to eliminate decode spikes.

Benchmark on RTX 3070 Ti (8GB VRAM)

- **Flux.1 Schnell (4 steps, 1024x1024): 18.4 seconds** per image (Peak VRAM: 7.2 GB).
- **Flux.1 Dev (20 steps, 1024x1024): 52.1 seconds** per image (Peak VRAM: 7.4 GB).

ComfyUI Launch Flags for 8GB VRAM

```
python main.py --lowvram --opt-sdp-attention --preview-method auto
```

MODULE 04 · THE $50 24/7 SOVEREIGN SERVER NODE (THINKPAD T440)

Never leave a 250W gaming GPU running 24/7 for cron jobs, background scrapers, or Telegram listeners.

Hardware Specifications

- **Machine:** Lenovo ThinkPad T440 (Used: ~$50 on eBay / DBA).
- **CPU:** Intel Core i5-4300U (2 cores, 4 threads, 15W TDP).
- **RAM:** 8 GB DDR3L.
- **Storage:** 256 GB SATA SSD.
- **OS:** Debian 12 Minimal (Headless).
- **Average Idle Power: 6.8 Watts** (~1.50 EUR / month in electricity).

Architecture

1. **Always-On Gateway:** ThinkPad runs `PM2` background daemons, SQLite task queues, and Telegram/Signal bots.
2. **Encrypted Mesh Networking:** `Tailscale` connects the ThinkPad, your phone, and the RTX 3070 Ti workstation across cellular/remote networks without port forwarding.
3. **Wake-on-LAN (WoL):** The ThinkPad monitors the task queue. When heavy generation is required, it triggers a Magic Packet over LAN to boot the RTX 3070 Ti, assigns the task, verifies completion, and shuts down the workstation.

MODULE 05 · ZERO-CLOUD AUTONOMOUS AGENT SWARM (OLLAMA + SQLITE)

Replace $200+/month multi-agent cloud frameworks (CrewAI/LangChain on OpenAI API) with an atomic, local Python queue.

SQLite3 Write-Ahead Logging (WAL) Schema

```
PRAGMA journal_mode=WAL;
CREATE TABLE IF NOT EXISTS agent_tasks (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    agent_name TEXT NOT NULL,
    task_type TEXT NOT NULL,
    payload TEXT NOT NULL,
    status TEXT DEFAULT 'pending',
    result TEXT,
    retry_count INTEGER DEFAULT 0,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    updated_at DATETIME DEFAULT CURRENT_TIMESTAMP
);
CREATE INDEX IF NOT EXISTS idx_status_agent ON agent_tasks(status, agent_name);
```

Production Python Task Dispatcher Daemon

```
import sqlite3, json, time, urllib.request

DB_PATH = "/var/local/agent_queue.db"
OLLAMA_API = "http://127.0.0.1:11434/api/generate"

def lease_next_task(agent_name):
    with sqlite3.connect(DB_PATH, timeout=10) as conn:
        cursor = conn.cursor()
        cursor.execute('''
            UPDATE agent_tasks
            SET status = 'processing', updated_at = CURRENT_TIMESTAMP
            WHERE id = (
                SELECT id FROM agent_tasks
                WHERE status = 'pending' AND agent_name = ?
                ORDER BY created_at ASC LIMIT 1
            )
            RETURNING id, payload;
        ''', (agent_name,))
        row = cursor.fetchone()
        return row if row else (None, None)

def worker_loop(agent_name, model):
    print(f"[*] Agent {agent_name} online. Monitoring queue...")
    while True:
        task_id, payload = lease_next_task(agent_name)
        if task_id:
            # Process with Ollama and commit status
            ...
        time.sleep(2)
```

✦ **VERIFICATION & SIGN-OFF:** This manual represents the real-world configuration powering lucaautomated.com. Zero cloud dependencies, zero subscription burn, sovereign compute.
Stefan Luca · Luca Automated AI Studio · https://lucaautomated.com