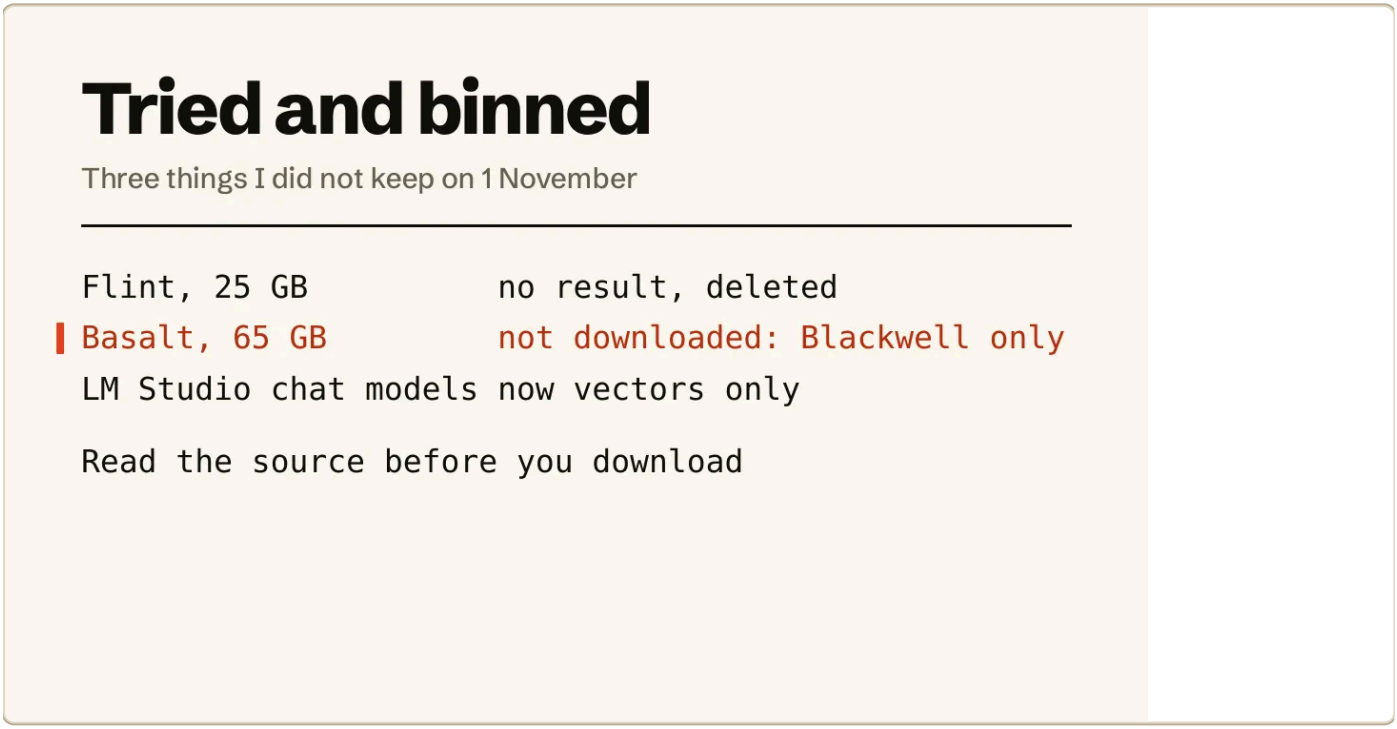
What I tried and binned, and what each one cost

Three local-model options I stopped using, with what each cost and the verdict: a 25 GB model that timed out in a job, a Blackwell-only fork ruled out from its source before a 65 GiB download, and the local rung's first night, which passed none of 11.



Published
1 November 2026

Reading time
3 minutes

About
Mullet

Tags
AI, Linux

More on: Mullet

Three local-model options I stopped using, with what each cost and the verdict. The Flint and Basalt verdicts judge my setup (my configuration and my card), not the models.

Flint: 25 GB, no result

- **What it is.** A mixture-of-experts coding model with 39.8 billion parameters and about 2.5 billion active, according to its model card. The vendor claims 74.2% on SWE-bench Verified. That is the vendor's figure and I have not checked it.
- **What I ran.** The Q4_K_XL file (25 GB) in LM Studio at 64k context, with 30% of the layers on the GPU. A short prompt gave a correct function at about 22 tokens a second. That speed is my measurement and I no longer have the timing log, so treat it as unchecked.
- **What it cost.** Two attempts in a Mullet job, no pass. Attempt 1 aborted after 48 turns with "The operation timed out". My notes record a request timing out at 5 minutes, with 692,529 tokens sent in by then (about 14,400 a turn on average,

arithmetic). Attempt 2 stopped at 40 turns. That is 88 turns of work (arithmetic) and nothing to merge.

- **Likely factor.** LM Studio's hardware config had the setting that puts the KV cache on the GPU switched off, which my notes say puts attention on the CPU. I did not test a fixed setup.
- **Strata.** My notes record Flint's model type as unsupported by Strata. I did not rerun that check, because the file is gone.
- **Verdict.** Binned, and I deleted the 25 GB.

Basalt: about 65 GiB, never downloaded

- **What it is.** A fork of Strata for NVIDIA Blackwell GPUs. Its model card lists a best measured 341 tokens a second on prose and 646 on structured output (the IQ3_S pack, single stream, on an RTX 5090 plus an RTX 5060 Ti), and says the numbers move with every release. Those are the project's numbers, not mine. The code is at https://github.com/jesdga95/basalt (https://github.com/jesdga95/basalt) and the model is at https://huggingface.co/jesdga/Qwen3.8-Flash-Next-Basalt (https://huggingface.co/jesdga/Qwen3.8-Flash-Next-Basalt).
- **What it cost.** Some reading and no disk. The packs on the model card start at 64.9 GiB, so I read the source before downloading anything. Three things rule out my card:
  - The build file stops with a FATAL_ERROR that says "basalt builds for compute capability 12.0 only".
  - The runtime's GPU check accepts only compute capability 12.0. My card is 8.9.
  - The build compiles FP4 block-scaled tensor-core kernels for the 120a variant of that architecture only. My card is Ada, which is not that.
- **Verdict.** Ruled out on this machine. I have not run Basalt, so I have nothing to say about its speed.

The local rung's first night

- **What it is.** The local rung of the delegation ladder, on its first night. My progress notes for 25 October record it at 0 of 11, with 7 timeouts. The notes do not say whether the 11 are steps or attempts. The local set then was Ornith 35B, Qwen3.6 35B and Ornith 9B, and I did not record which model took each one.
- **Verdict.** The local rung passed none that night, and I can't say more than that. LM Studio still holds those chat models and two small embedding models. I now keep it only for local vectors, which is what the embedding models are for.

What I would do first

Read the build file and the runtime checks, then the memory the model needs and the hardware it assumes. Download last. A short read of the source or the numbers costs little. A 25 GB or 65 GiB download and a trial run cost far more.

None of this proves those models are bad. Flint might work well on a better setup, and Basalt is built for a card I do not own. I can only say what I measured, which is that they did not work here.

The code is at https://github.com/tomgould/mullet-public (https://github.com/tomgould/mullet-public).