mmind.ai

MODEL GUIDE · JULY 2026

Kimi K3

The Agentic Engineering Guide

2.8T parameters **1M** token context **KDA + AttnRes** architecture

16 / 896 experts active **2.5×** scaling efficiency vs. K2

Moonshot AI's flagship open-weight model — what it is, how it's built, and how to build agents on it.

projekte@mmind.ai
mmind.ai

OVERVIEW

What is Kimi K3

Kimi K3 is Moonshot AI's most capable model to date — a 2.8-trillion-parameter flagship, and the first open-source model to reach the 3-trillion-parameter class. It's built for frontier, agentic use: long-horizon coding, knowledge work, and reasoning, with native visual understanding and a 1-million-token context window.

K3's API and apps went live in mid-July 2026; full open model weights are scheduled for release by **July 27, 2026**, alongside a complete technical report. Moonshot is currently coordinating with inference partners and open-source maintainers to align technical details before that release.

2.8T
total parameters — first open model in the ~3T class

1,048,576
token context window (1M), flat-priced, no tiering

16 / 896
experts active per token (Stable LatentMoE)

2.5×
overall scaling efficiency vs. Kimi K2

9 / 12
months (Jul 2025–Jul 2026) Kimi held the open-source scale frontier

Native
vision understanding — images and video, in the same agent loop

Why it matters: K3 is open-weight, not a closed API-only release. Every parameter will be inspectable and self-hostable — the same "democratize intelligence" positioning Moonshot has held since K2, now at flagship scale.

ARCHITECTURE

Under the hood: KDA + Attention Residuals

In mid-2025, Moonshot researchers previewed two experimental directions at a technical talk: a linear-attention variant for long context, and an unnamed "attention applied across layers" mechanism for depth. Both have now shipped as production architecture in K3.

- **Kimi Delta Attention (KDA)**
  A hybrid linear-attention mechanism, evolved from Moonshot's earlier "Kimi Linear" work. KDA mixes efficient linear-attention layers with full-attention layers so information keeps flowing cleanly across very long sequences — the backbone of K3's 1M-token context.

- **Attention Residuals (AttnRes)**
  Replaces the classic residual connection with an attention operation applied across layers instead of tokens — each layer can attend over all previous layers' hidden states, not just the one before it. Helps very deep networks train more stably and use compute more efficiently.

- **Stable LatentMoE**
  A sparser mixture-of-experts framework: K3 activates only 16 of its 896 experts per token. Combined with KDA, AttnRes, and improved training methodology and data recipes, this gives K3 roughly **2.5× the overall scaling efficiency of K2** — more capability per unit of compute, not just more parameters.

> **The throughline:** KDA and AttnRes were research previews a year before K3 shipped. Architecture choices Moonshot flags as experimental are worth tracking — they tend to become next year's flagship.

CAPABILITIES

What K3 is good at

Coding

Strong long-horizon coding: with minimal human supervision, K3 sustains long-running engineering tasks, understands large codebases, and coordinates terminal tools across a session. It also combines software engineering with visual reasoning — using screenshots and visual feedback to iterate on frontend work, game development, and CAD.

Knowledge work

Beyond public benchmarks, Moonshot reports consistent gains for K3 (max) on internal evaluations built from recurring patterns in real user-agent collaboration. The claim: broad improvement in agentic knowledge-work capability, not just narrow benchmark tuning.

> **Benchmarks (Moonshot-reported, "max" reasoning effort):** strong results on long-horizon coding suites — DeepSWE, Terminal-Bench, SWE Marathon, kernel optimization, full compiler development, chip design. Tops the Frontend Code Arena leaderboard; competitive with Claude Fable 5 and ahead of many other current models.

These figures come from Moonshot's own technical blog and docs. Full weights and the complete technical report are still pending (by July 27, 2026) — treat benchmark numbers as Moonshot-reported until independently reproduced.

API

Getting started

K3 is OpenAI SDK-compatible. Get an API key at **platform.kimi.ai/console/api-keys**, or try it first in the **Playground**. Point the standard OpenAI client at Moonshot's endpoint and call the model as kimi-k3.

```
import os
from openai import OpenAI

client = OpenAI(
  api_key=os.environ["MOONSHOT_API_KEY"],
  base_url="https://api.moonshot.ai/v1",
)

completion = client.chat.completions.create(
  model="kimi-k3",
  reasoning_effort="max",
  messages=[{"role": "user", "content": "Introduce Kimi K3 in one sentence."}],
)
print(completion.choices[0].message.content)
```

Pricing is flat pay-as-you-go — no tiering by context length, even at 1M tokens:

Item	Price
Input — cache hit	$0.30 / MTok
Input — cache miss	$3.00 / MTok
Output	$15.00 / MTok

Key parameters to know:

reasoning_effort
low / high / max (default max). K3's thinking mode is always on — this dial controls how much.

max_completion_tokens
Defaults to 131,072; can be set up to 1,048,576.

temperature / top_p / n / penalties
Fixed at 1.0 / 0.95 / 1 / 0 — omit them from requests entirely.

vision input
Base64 or ms://<file-id> only — no public image URLs; content must be an array of objects.

AGENTIC ENGINEERING

Building agents on K3

1. **Tool calling, done completely**
   Use tool_choice="required" to force a first tool call. After executing each call, append the **complete** assistant message plus one tool-result message per call — never trim it down to just the text content.

2. **Dynamic tool loading**
   Inject a full tool definition mid-conversation via a content-less system message. It takes effect from that point in the transcript — but the server doesn't retain it, so keep it in your own message history.

3. **Official tools via Formula**
   Moonshot's managed tool layer: fetch tool definitions from Formula's /tools endpoint, add them to your request, and execute returned calls via /fibers. Note: web search is being reworked and isn't recommended in production yet.

4. **Vision in the agent loop**
   Feed screenshots or short videos directly into the same tool-using agent — not through a separate captioning step. This is what makes visual-feedback workflows (UI review, CAD, game dev, frontend iteration) work natively.

5. **1M context + automatic caching**
   No manual cache IDs or TTLs. Keep a long prefix (a codebase dump, a knowledge base) stable across calls and later requests automatically attempt a cache hit — at roughly 1/10th the input price.

6. **Reasoning effort as a per-call dial**
   Set reasoning_effort per request, not per deployment: low for cheap routine calls, max for the hard step in a long agent trajectory.

7. **Agent swarms for parallel work**
   Kimi's apps expose multi-sub-agent orchestration: an orchestrator spawns parallel sub-agents for independent sub-tasks. Worth adopting for research- or report-style workloads — but only where sub-tasks are genuinely parallelizable.

FIELD NOTES

Five design principles

1

Feed it your real environment, not a summary

1M native context plus native vision means far less pressure to pre-chunk or build a RAG layer around K3. Test the native long-context path before you build retrieval infrastructure to work around a limit that may no longer exist.

2

Design for generalization, not your tool list

K3 is built as a horizontal, general-purpose agent. Prefer clear tool descriptions and prompting over heavy fine-tuning on your exact schema — narrow fine-tuning is the fastest way to lose the generalization you're paying for.

3

Track completion, not just delegation

If you build multi-agent or swarm orchestration on top of K3, monitor how many spawned sub-tasks actually finish — not just how many were started. A busy-looking orchestrator log can hide an agent that never converges.

4

Treat reasoning effort as a cost dial, not a fixed setting

Vary reasoning_effort call by call. Most of an agent trajectory is routine; only a few steps need the model thinking at maximum depth.

5

Re-verify after July 27, 2026

Full open weights and the complete technical report are still landing. Benchmark numbers in this guide are Moonshot's own until independently reproduced — check back once the weights and report are public.

REFERENCES

Sources

- **Kimi K3 — introduction & quickstart**
  platform.kimi.ai/docs/guide/kimi-k3-quickstart
- **Kimi K3 — technical blog (benchmarks & case studies)**
  www.kimi.com/blog/kimi-k3
- **Kimi models overview**
  platform.kimi.ai/docs/models
- **Kimi K3 pricing**
  platform.kimi.ai/pricing/chat-k3
- **Playground / API keys**
  platform.kimi.ai/playground · platform.kimi.ai/console/api-keys
- **Tool calling, dynamic tool loading, official tools (Formula), context caching, vision, structured output, partial mode, tool-calling best practice**
  platform.kimi.ai/docs/guide/*
- **Full documentation index**
  platform.kimi.ai/docs/llms.txt

Release-status verification and benchmark cross-references compiled via independent research (Grok DeepSearch, July 2026) against Moonshot AI's own primary sources. Confidence: high for release status and API specifics; benchmark specifics and full architecture detail remain pending Moonshot's complete technical report.

MMIND.ai
AI strategy & agentic engineering for organizations building with frontier open models.

projekte@mmind.ai
mmind.ai