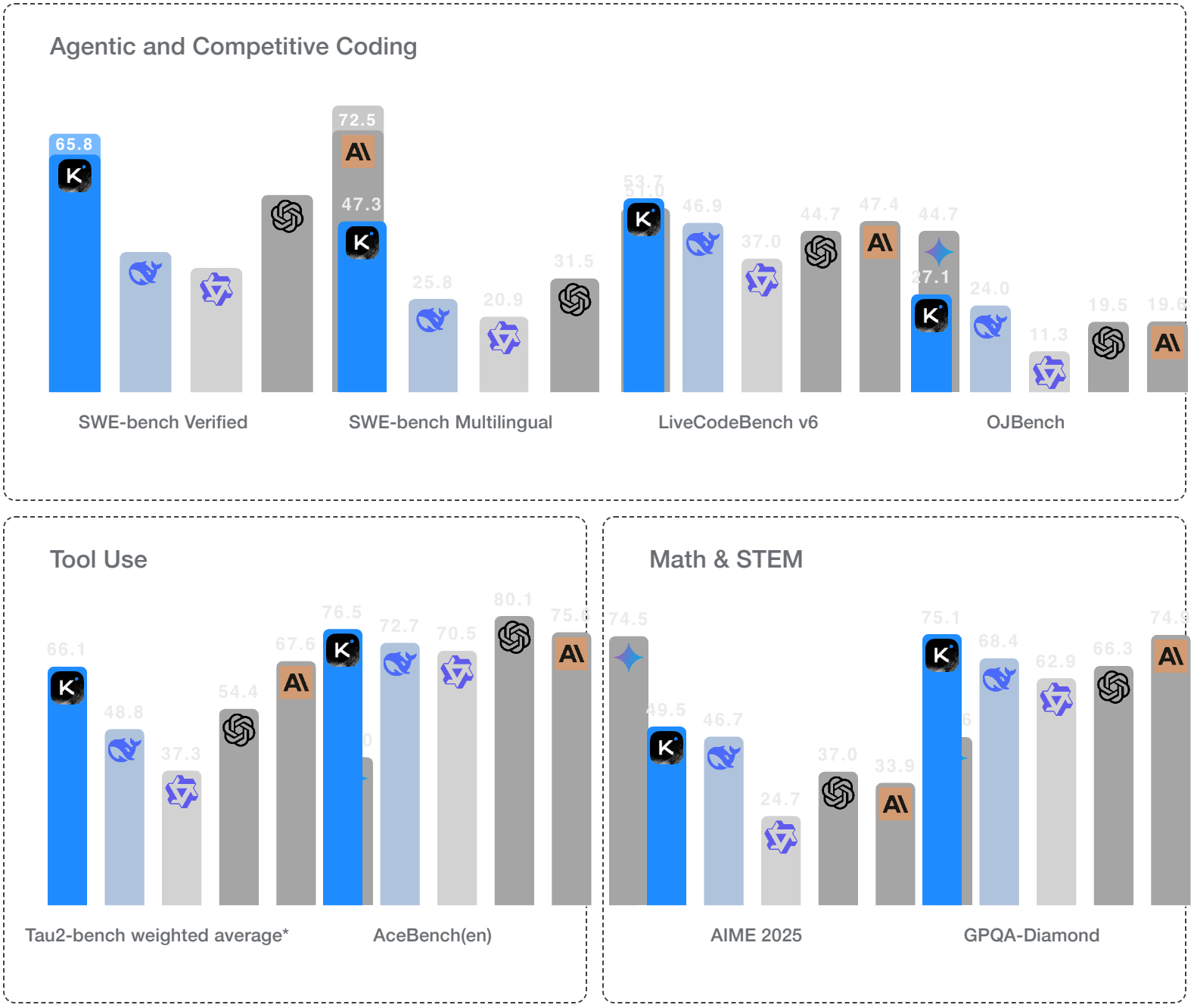
Kimi K2: Open Agentic Intelligence

Kimi K2 is our latest Mixture-of-Experts model with 32 billion activated parameters and 1 trillion total parameters. It achieves state-of-the-art performance in frontier knowledge, math, and coding among non-thinking models. But it goes further — meticulously optimized for agentic tasks, Kimi K2 does not just answer; it acts.

And now, it is within your reach. Today, we are open-sourcing:

- **Kimi-K2-Base:** The foundation model, a strong start for researchers and builders who want full control for fine-tuning and custom solutions.
- **Kimi-K2-Instruct:** The post-trained model best for drop-in, general-purpose chat and agentic experiences. It is a reflex-grade model without long thinking.

With Kimi K2, advanced agentic intelligence is more open and accessible than ever. We can't wait to see what you build.



- All models evaluated above are non-thinking models.
- For Tau2-Bench, average is weighted by tasks.

- For Swe-Bench Multilingual, we evaluated only Claude 4 Sonnet because the cost of Claude 4 Opus was prohibitive.

Use cases

Give Kimi K2 your tools and describe your task.

It automatically understands how to use the tools and gets the job done. You don't have to write any complex workflow for agentic applications.

Using 2020–2025 salary data, test the effect of remote-work ratio on salary a nd determine whether this effect differs significantly across experience levels (EN / MI / SE / EX)—that is, whether an interaction effect exists. Support your analysis with statistical evidence and rich visualizations. Use a consistent, har monious color palette (e.g., pastel, muted) across all plots.



To begin, I'll load the dataset you've provided to understand its structure and contents. This initial step is crucial for planning the subsequent analysis of how the remote-work ratio affects salary across different experience levels.

Use ipython Task completed

I will now filter the dataset to include only the years 2020 through 2025. Following that, I'll create violin plots to visualize the distribution of salaries for each experience level, segmented by their remote-work ratio. This initial visualization will provide a clear, intuitive understanding of the data before I proceed with a more rigorous statistical analysis.

Use ipython Task completed

To prepare for a more rigorous statistical analysis, I will now categorize the remote_ratio into distinct groups: 'On-site', 'Hybrid', and 'Remote'. This transformation will make it easier to interpret the results. I will then generate a box plot using these new categories to visually compare the salary distributions across the different experience levels. This will help in clearly illustrating any potential interaction effects.

Use ipython Task completed

Now that I've categorized the data, I'll proceed with the statistical analysis. I'm going to run a two-way ANOVA to test for both the main effects of the remote-work category and experience level, as well as their interaction. This will provide us with concrete statistical evidence to determine if the relationship between remote work and salary changes significantly across different experience levels.

Use ipython Task completed

It appears the statsmodels library isn't available in this environment. As an alternative, I will use the scipy.stats library to perform a two-way ANOVA. This will allow me to test

Imagine using Kimi K2 to explore remote-work salaries with the Salary Data Analysis example, where 16 IPython calls generate stats, visualizations, and an interactive webpage of insights. Dive into the Stanford NLP Genealogy, and get Kimi K2 to generate an interactive site built through 5 web searches, 4 browsings, 3 clicks, 5 scrolls, 6 edits, and 2 deployments. Or plan your dream Coldplay Tour 2025 in London with Kimi K2, it crafted the plan for you through 17 seamless tool calls spanning search, calendar, Gmail, flights, Airbnb, and restaurant bookings.

Bring Kimi K2 to your command line. It edits files. It runs commands.

Kimi K2 understands your environment, decides what actions to take, and executes them seamlessly.

```
- Configuring camera and renderer settings
- Adding lighting system
- Implementing render loop
```

Salary Data Analysis

Example 1 Example 2 Example 3

Chat Result

```
- Drawing geometric patterns (lines, circles, angles)
- Implementing unique block textures:
  - Grass texture generation
  - Dirt texture generation
  - Stone texture generation
```

For clarity, the terminal displays only a summary of each trajectory. In every example, Kimi K2 orchestrates multiple tools and commands behind the scenes to accomplish complex objectives. For instance, Kimi K2 can automate Minecraft development in JavaScript: it manages rendering, runs and debugs test cases, captures logs on failure, and iteratively improves the code until all tests succeed. For pre/post norm analysis, Kimi K2 uses the Weights & Biases (wandb) data reader to extract insights from language model experiments and generates a polished analysis report. When converting a Flask project to Rust, Kimi K2 systematically refactors the codebase and runs performance benchmarks to ensure robust results.

Benchmarking Kimi K2

Evaluation Results

Kimi-K2-Instruct Kimi-K2-Base

The table below details the performance of Kimi-K2-Instruct, showing that it matches—or outperforms—the latest open-source and proprietary models across a diverse set of tasks. The model shines on knowledge-intensive and reasoning benchmarks, delivering outstanding results in natural-language understanding, mathematics and sciences, code generation, and agentic tool uses.

Terminal Result

		Open Source			Proprietary		
Benchmark	Metric	Kimi-K2-Instruct	DeepSeek-V3-0324	Qwen3-235B-A22B (Non-thinking)	Claude Sonnet 4 (w/o extended thinking)	Claude Opus 4 (w/o extended thinking)	GPT-4.1
Coding Tasks							
LiveCodeBench v6 (Aug 24-May 25)	Pass@1	**53.7**	46.9	37.0	48.5	47.4	44.7
OJBench	Pass@1	**27.1**	24.0	11.3	15.3	19.6	19.5
MultiPL-E	Pass@1	**85.7**	83.1	78.2	88.6	**89.6**	86.7

		Open Source			Proprietary		
Benchmark	Metric	Kimi-K2-Instruct	DeepSeek-V3-0324	Qwen3-235B-A22B (Non-thinking)	Claude Sonnet 4 (w/o extended thinking)	Claude Opus 4 (w/o extended thinking)	GPT-4.1
SWE-bench Verified (Agentless Coding)	Single Patch without Test (Acc)	**51.8**	36.6	39.4	50.2	**53.0**	40.8
SWE-bench Verified (Agentic Coding)	Single Attempt (Acc)	**65.8**	38.8	34.4	**72.7***	72.5*	54.6
	Multiple Attempts (Acc)	**71.6**	—	—	**80.2***	79.4*	—
SWE-bench Multilingual (Agentic Coding)	Single Attempt (Acc)	**47.3**	25.8	20.9	**51.0**	—	31.5
TerminalBench	Inhouse Framework (Acc)	**30.0**	—	—	35.5	**43.2**	8.3

		Open Source			Proprietary		
Benchmark	Metric	Kimi-K2-Instruct	DeepSeek-V3-0324	Qwen3-235B-A22B (Non-thinking)	Claude Sonnet 4 (w/o extended thinking)	Claude Opus 4 (w/o extended thinking)	GPT-4.1
	Terminus (Acc)	**25.0**	16.3	6.6	—	—	**30.3**
Aider-Polyglot	Acc	60.0	55.1	**61.8**	56.4	**70.7**	52.4
Tool Use Tasks							
Tau2 retail	Avg@4	**70.6**	69.1	57.0	75.0	**81.8**	74.8
Tau2 airline	Avg@4	**56.5**	39.0	26.5	55.5	**60.0**	54.5
Tau2 telecom	Avg@4	**65.8**	32.5	22.1	45.2	57.0	38.6
AceBench	Acc	**76.5**	72.7	70.5	76.2	75.6	**80.1**
Math & STEM Tasks							
AIME 2024	Avg@64	**69.6**	59.4*	40.1*	43.4	48.2	46.5
AIME 2025	Avg@64	**49.5**	46.7	24.7*	33.1*	33.9*	37.0

		Open Source			Proprietary		
Benchmark	Metric	Kimi-K2-Instruct	DeepSeek-V3-0324	Qwen3-235B-A22B (Non-thinking)	Claude Sonnet 4 (w/o extended thinking)	Claude Opus 4 (w/o extended thinking)	GPT-4.1
MATH-500	Acc	**97.4**	94.0*	91.2*	94.0	94.4	92.4
HMMT 2025	Avg@32	**38.8**	27.5	11.9	15.9	15.9	19.4
CNMO 2024	Avg@16	74.3	**74.7**	48.6	60.4	57.6	56.6
PolyMath-en	Avg@4	**65.1**	59.5	51.9	52.8	49.8	54.0
ZebraLogic	Acc	**89.0**	84.0	37.7*	79.7	59.3	58.5
AutoLogi	Acc	**89.5**	88.9	83.3*	**89.8**	86.1	88.2
GPQA-Diamond	Avg@8	**75.1**	68.4*	62.9*	70.0*	74.9*	66.3
SuperGPQA	Acc	**57.2**	53.7	50.2	55.7	56.5	50.8
Humanity's Last Exam (Text Only)	Acc	4.7	5.2	**5.7**	5.8	**7.1**	3.7

		Open Source			Proprietary		
Benchmark	Metric	Kimi-K2-Instruct	DeepSeek-V3-0324	Qwen3-235B-A22B (Non-thinking)	Claude Sonnet 4 (w/o extended thinking)	Claude Opus 4 (w/o extended thinking)	GPT-4.1
General Tasks							
MMLU	EM	**89.5**	89.4	87.0	91.5	**92.9**	90.4
MMLU-Redux	EM	**92.7**	90.5	89.2*	93.6	**94.2**	92.4
MMLU-Pro	EM	81.1	**81.2***	77.3	83.7	**86.6**	81.8
IFEval	Prompt Strict	**89.8**	81.1	83.2*	87.6	87.4	88.0
Multi-Challenge	Acc	**54.1**	31.4	34.0	46.8	49.0	36.4
SimpleQA	Correct	**31.0**	27.7	13.2	15.9	22.8	**42.3**
Livebench (2024/11/25)	Pass@1	**76.4**	72.4	67.6	74.8	74.6	69.8

- All models evaluated above are non-thinking models.
- Bold denotes global SOTA, and underlined denotes open-source SOTA.
- Data points marked with * are taken directly from the model's tech report or blog.

- All metrics, except for SWE-bench Verified (Agentless), are evaluated with an 8k output token length. SWE-bench Verified (Agentless) is limited to a 16k output token length.
- Kimi K2 achieves 65.8% pass@1 on the SWE-bench Verified tests with bash/editor tools (single-attempt patches, no test-time compute). It also achieves a 47.3% pass@1 on the SWE-bench Multilingual tests under the same conditions. Additionally, we report results on SWE-bench Verified tests (71.6%) that leverage parallel test-time compute by sampling multiple sequences and selecting the single best via an internal scoring model.
- To ensure the stability of the evaluation, we employed avg@k on the AIME, HMMT, CNMO, PolyMath-en, GPQA-Diamond, EvalPlus, Tau2.
- Some data points have been omitted due to prohibitively expensive evaluation costs.

Open Agentic Intelligence

Pre-training is the crucial foundation for Agentic Intelligence, establishing the priors that makes reinforcement learning (RL) exploration tractable, efficient, and generalizable. However, as Ilya Sutskever also observes, human data is a finite "fossil fuel", and its growth is lagging far behind the pace of compute. This makes **token efficiency** during pre-training a new critical coefficient in the AI scaling laws.

Post-training is pivotal in the "Era of Experience" (David Silver, Richard Sutton, 2025). In this era, LLMs increasingly learn from their own self-generated interactions, receiving rewards that free them from the limits of human data and enable them to surpass human capabilities.

Kimi K2 is forged from these very insights.

MuonClip Optimizer

Without rigor, given an approximately finite pretraining dataset and a fixed model configuration, a more token-efficient optimizer generates more intelligence. Our previous work Moonlight has demonstrated that the Muon optimizer substantially outperforms the widely-used AdamW optimizer for LLM training.

Kimi K2 was designed to further scale up Moonlight, which employs an architecture similar to DeepSeek-V3. Based on scaling-law analysis, we reduce the number of heads for long-context efficiency, and increase MoE sparsity for greater token efficiency. While scaling up, we encountered a persistent challenge: training instability caused by exploding attention logits, an issue that occurs more frequently with Muon but less with AdamW in our experiments. Existing solutions such as logit soft-capping and query-key normalization were found inadequate.

To address this, we introduce the MuonClip optimizer that improves Muon with our proposed qk-clip technique. Specifically, qk-clip stabilizes training by directly rescaling the weight matrices of the query and key projections after Muon updates, thus controlling the scale of attention logits at the source. Concretely, the query and key projections are scaled as follows:

$$q_i = \eta^{\alpha} W_q x_i$$

$$k_i = \eta^{1-\alpha} W_k x_i$$

where α is a balancing hyperparameter, so the attention logit becomes:

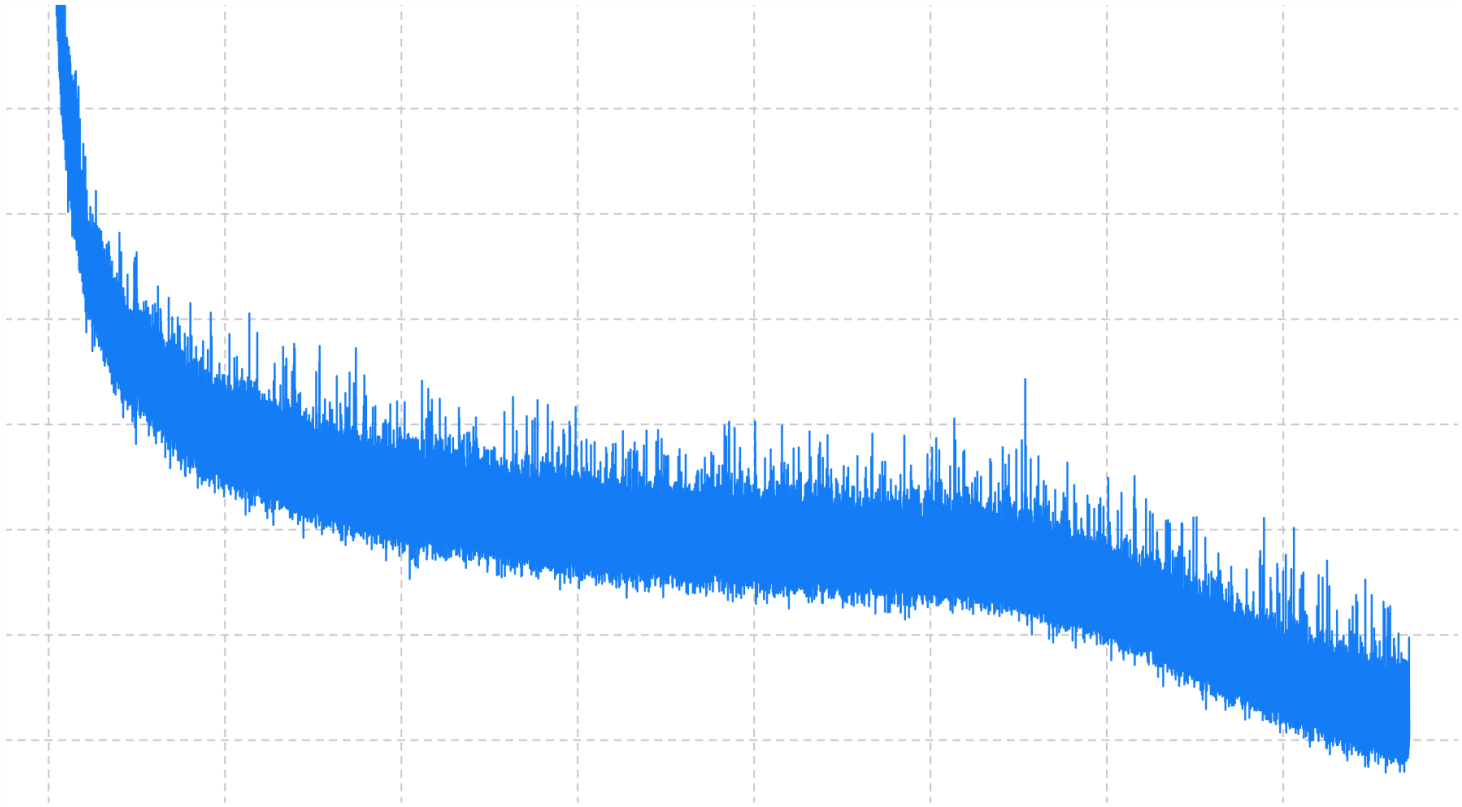
$$(\eta^{\alpha} q_i)^{\top} (\eta^{1-\alpha} k_j) = \eta\, q_i^{\top} k_j$$

The adaptive factor η (with threshold t) is set after every step based on the max attention logit in this step:

$$\eta = \min\left(\frac{t}{\max_{i,j}\left(q_i^{\top} k_j\right)}, 1\right)$$

where t is a pre-set threshold. This is a general technique that can be possibly applied to other stabilization use cases.

Our experiments show that MuonClip effectively prevents logit explosions while maintaining downstream task performance. In practice, Kimi K2 was pre-trained on 15.5T tokens using MuonClip with zero training spike, demonstrating MuonClip as a robust solution for stable, large-scale LLM training.

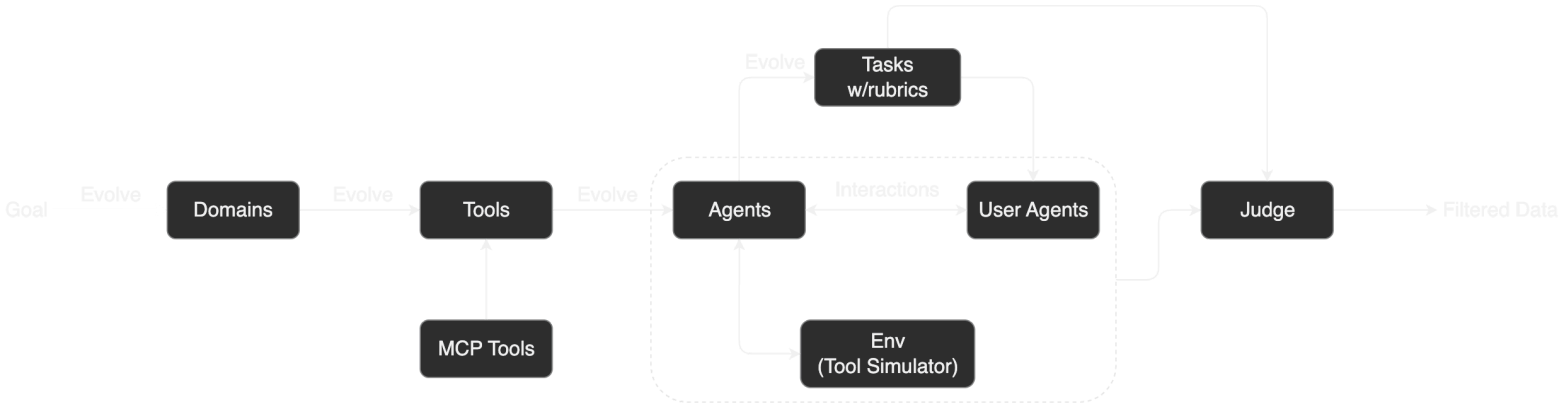


Agentic Capabilities

The enhanced agentic capabilities of Kimi K2 originate from two important aspects — large-scale agentic data synthesis and general reinforcement learning.

Large-Scale Agentic Data Synthesis for Tool Use Learning: To teach the model sophisticated tool-use capabilities, we developed a comprehensive pipeline inspired by ACEBench that simulates real-world tool-using scenarios at scale. Our approach systematically evolves hundreds of domains containing thousands of tools—including both real MCP (Model Context Protocol) tools and synthetic ones—then generates hundreds of agents with diverse tool sets.

All tasks are rubric-based, enabling consistent evaluation. Agents interact with simulated environments and user agents, creating realistic multi-turn tool-use scenarios. An LLM judge evaluates simulation results against task rubrics, filtering for high-quality training data. This scalable pipeline generates diverse, high-quality data, paving the way for large-scale rejection sampling and reinforcement learning.



General Reinforcement Learning: The key challenge is to apply RL to tasks with both verifiable and non-verifiable rewards; typical examples of verifiable tasks are math and competition coding, while writing a research report is usually viewed as non-verifiable. Going beyond verifiable rewards, our general RL system uses a self-judging mechanism where the model acts as its own critic, providing scalable, rubric-based feedback for non-verifiable tasks.

Meanwhile, on-policy rollouts with verifiable rewards are used to continuously update the critic so that the critic keeps improving its evaluation accuracy on the latest policy. This can be viewed as a way of using verifiable rewards to improve the estimation of non-verifiable rewards.

Getting started with Kimi K2

Try Kimi K2 on kimi.com

Starting today, Kimi users on web and mobile can select and use the new Kimi K2 model for free. At this moment, our MCP features for web and app are still in development. We hope to begin rolling them out in the coming weeks. In the meantime, you're welcome to try our Researcher for an early look at its agentic capabilities. Please note that vision features are not supported for Kimi K2 yet.

Use Kimi K2 with API

The Kimi Platform offers an OpenAI/Anthropic compatible interface, allowing for easy adaptation of your existing applications to Kimi K2. We encourage developers to explore our tool calling API for building agent applications. For detailed information, visit platform.moonshot.ai.

Serve Kimi K2 on your own

We recommend running Kimi K2 on one of the following inference engines: vLLM, SGLang, KTransformers, or TensorRT-LLM. For detailed deployment instructions, please see our GitHub repository.

What's next

While Kimi K2 serves as a strong foundation for open agentic intelligence, a general agent uses more advanced capabilities such as thinking and visual understanding. We plan to add these to Kimi K2 in the future.

Limitations

In our internal tests, we've identified some limitations in current Kimi K2 models. When dealing with hard reasoning tasks or unclear tool definition, the model may generate excessive tokens, sometimes leading to truncated outputs or incomplete tool calls. Additionally, performance may decline on certain tasks if tool use is enabled. When building complete software projects, one-shot prompting yields performance degradation compared to using K2 under an agentic framework. We are working to address these issues in future releases and looking forward to more feedbacks.