

Alibaba releases Qwen3.8-Max, 2.4 trillion parameters, and promises its weights

Source copy kept by wheresthe.ai for the story "Alibaba releases Qwen3.8-Max, 2.4 trillion parameters, and promises its weights" (<https://wheresthe.ai/d/cn-alibaba-qwen-3-8-max-open-2026-08/>).

Source: <https://qwen.ai/blog?id=qwen3.8>

Text saved 18 Oct 2026 (the page text through a reader, because the site refused a direct fetch).

SHA-256 of the saved text: 5735ea1b2793343dde8577aef120c60efcc5df5440a2776b2e521c9cbce43adb

QWEN STUDIO DISCORD

Today, we are officially releasing **Qwen 3.8-Max**, the most capable model in the Qwen family to date. This also marks the first time we will open-source the weights of a Qwen-Max-class model — the open weights will be released next week. Built upon the architectural foundation of Qwen 3.5, Qwen 3.8-Max scales to **2.4 trillion** parameters, delivering comprehensive improvements across coding, work, research, and long-horizon tasks. It can not only answer more challenging questions, but also complete complex tasks end-to-end with greater reliability, producing dependable deliverables.

Qwen 3.8-Max — now available via QwenCloud: **2.4T** parameters (95B active), with open weights releasing next week **comprehensive improvements across coding, work, research, and long-horizon tasks** **end-to-end and dependable delivery of complex tasks**

Call via API on QwenCloud.

Coding

For a top model, coding today means far more than writing a function on request — it means taking a real, multi-day project from an empty folder all the way to a finished result, on its own. We tested Qwen3.8-Max on three such challenges, where every result had to be earned by actually writing and running code, with **no human help at all**. One thread runs through all three: Qwen3.8-Max doesn't just follow a fixed plan — it **self-evolves through feedback loops**, whether that means building a harness that upgrades itself, refining a research method experiment after experiment, or climbing a competition leaderboard submission after submission.

10+ Days of Autonomous Coding: Building a Self-Evolving Harness

In this case, Qwen3.8-Max was asked to create the `oh-my-cli` project from scratch and, over a 10+ day long-horizon autonomous coding run, build a self-evolving harness. It brings user feedback, advanced community practices, and the model's own self-test results into one engineering loop: requirements are normalized into issues, automatically claimed and executed by agents, and continuously iterated through code, tests, previews, and logs. The complete project trace is publicly available in the GitHub repository `qwen-code-dev-bot/oh-my-cli`.

Key implementation details in the autonomous coding harness:

Loop Engineering Setup: task state, dispatch, and recovery. Qwen3.8-Max combines an issue state machine, dispatcher, monitor, and watchdog into one execution loop: after a new requirement enters GitHub Issues, an agent claims it through the state machine and moves through `ready → leased → active`; once implementation is complete, E2E tests and CI checks are triggered, and the PR is merged after passing. **Self-testing:** product self-testing and maintenance. After each update, the model triggers Build, Unit Test, E2E, and Desktop Lifecycle validation; abnormal states are routed back to the relevant issue / PR for fixes and re-verification. **Multi-source Evolution:** product upgrades from multiple demand signals. By converting community experience and user / developer feedback into executable work, the harness continuously evolves `/goal`, `/resume`, Dynamic Workflow, Session Replay, Desktop, and other capabilities.

As of **July 30, 2026**, after approximately **16 days** of fully autonomous AI operation, the repository had accumulated **265 commits**, **127 PRs**, and **151 issues**, demonstrating a continuously evolving autonomous coding capability.

Video 1. In a 10+ day long-horizon autonomous coding run, Qwen3.8-Max autonomously builds a self-evolving harness, continuously completing community requirement collection, issue dispatch, code generation, verification,

and self-repair.

Reproduce a research paper — then improve it

We handed Qwen3.8-Max a recent research paper — “Unified Data Selection for LLM Reasoning” — and asked it to: **reproduce the paper’s experiment in code, then try to do better.** The paper tackles a very practical question in AI training: when you have far more data than you can afford to train on, which examples are actually worth keeping? The paper’s answer is to prize the examples full of **“hard decision points”** — the moments in a worked solution where the model was genuinely unsure which way to go next.

The catch: Qwen3.8-Max started from **nothing but the paper and a set of GPUs** — no starter code, no ready-made pipeline. The data-processing scripts, the training code, the evaluation setup — it had to design and write **all of it from scratch**, exactly the kind of work that takes skilled engineers days.

Working **completely on its own for about five days** (~125 hours of continuous effort), Qwen3.8-Max wrote roughly **7,600 lines of code**, took over **1,100 actions**, and ran **33 rounds of GPU training**. It first spent ~37 hours rebuilding the paper’s full pipeline from zero and **reproduced its six main findings** — repeatedly fine-tuning a Qwen3-8B model on the data it selected and confirming the gains on hard math benchmarks (for instance, the paper’s selection method beats picking data at random by **+7.7%** on AIME24).

Then it went further, turning reproduction into **self-evolution**. Over the next ~88 hours it ran a self-improving research loop — form a hypothesis → write the code → run it on GPUs → analyze → try again — inventing and testing **18 improvement ideas of its own across four rounds**. Each round’s results fed the next round’s hypotheses, and by diagnosing what went wrong with each attempt it finally evolved a new method that **beats the paper’s own approach**, a **+2.7-point** gain on the competition-level math benchmark AIME24.

How the improvement search unfolded — 4 rounds, 18 ideas | Round | Best idea that round | Score (AIME24) | Gain vs. baseline | | --- | --- | --- | --- | | --- | Paper’s method, reproduced (baseline) | 49.58% | — | | 1 | Split the data by difficulty before selecting | 50.42% | +0.84 | | 2 | Weight examples by an entropy–score gap | 51.67% | +2.09 | | 3 | Tune the selection width | 51.25% | +1.67 | | 4 | **Count the hard decision points (“nhighgate”)** ★ | **52.29%** | **+2.71** |

Beat hundreds of human teams in 24 hours

Next we entered Qwen3.8-Max into a real online contest — the WWW2025 Multimodal Dialogue Intent Recognition Challenge, hosted on **Alibaba Cloud’s Tianchi platform**, where **526 human teams** were competing. The task: read customer-service chats — both the text and the screenshots — and correctly work out what the customer wants.

Working entirely on its own and under a strict **24-hour** time limit, Qwen3.8-Max read the competition rules and built a full solution in code. For the text side, it fine-tuned and ensembled several Chinese language models — **BERT, MacBERT, and RoBERTa**; for the product screenshots, it fine-tuned a vision-language model, **Qwen2.5-VL-7B**, backed by a **Chinese-CLIP** model for images its main model was unsure about. It then fused all of them into a single **weighted-voting system**, calibrating how much each model’s vote should count through cross-validation and adding extra image voters to break ties. Across **45 submissions** — each round’s feedback steering the next round of fine-tuning and re-weighting — its accuracy climbed steadily from **0.60** to a final **0.853**, beating **458 of the 526 human teams (87% of the field)**.

Together, these three cases show what makes Qwen3.8-Max stand out: it can stay focused on a hard, open-ended goal for days, come up with its own ideas, and turn them into working results — all without a human in the loop.

Work

Alongside coding, **real work** - the messy, multi-step, tool-heavy tasks that fill the working day in nearly every profession - is the other main track where frontier models create enormous economic value. Making Qwen3.8-Max broadly competent and reliably robust across these workflows is therefore central to our mission.

Scaling Real-World RL Systems. By jointly scaling RL environments and compute, we lift **general working competence** uniformly across several popular harnesses (QwenWork / Claude Code / Codex / OpenClaw / Hermes). Achieving this required addressing three coupled challenges:

1. **Continuously scaling decoupled real environments** along independent axes — Task (single-task → multi-

task → multi-day), `_Workspace_` (multi-file → hierarchical folders → complex heterogeneous folders), and `_Harness_` (category, version, skills) — so environment growth **compounds combinatorially** rather than requiring bespoke integration.

2. **A Universal Reward System** that internalizes heterogeneous verification — spanning execution-based checking, rubric-conditioned adjudication over text and rendered visual output, and agentic inspection — under automatically scalable rubrics. By unifying these modalities within **one reward system**, it provides a coherent and reliable source of reward across all environments, eliminating the inconsistency inherent in maintaining task-specific verifiers.

3. **An online data balancer** that shapes every batch to keep its distribution over tasks, difficulty, workspaces, and harnesses highly balanced, **suppressing inter-batch gradient variance** and thereby sustaining stable, continued scaling of RL compute.

Together these supply **breadth, reliable reward, and stability** — turning joint environment-and-compute scale into a measurable, horizontal lift in real-world working ability.

Fig 1. Qwen3.8-Max shows steady, consistent gains across dozens of in-house and public working benchmarks as RL training continues to scale up.

Fig 2. Qwen3.8-Max achieves comparable performance across many harnesses, including QwenWork, Claude Code, Codex, OpenClaw, and Hermes.

Testing the `_Breadth_` of Working Ability Across `_Hundreds_` of High-Value Professions

As frontier models take on an ever-widening role in economically valuable work, we stress-tested the **breadth** of Qwen3.8-Max's ability to deliver production-quality results in real workflows — spanning high-frequency tasks across **several hundred high-economic-value professions**. A few representative showcases:

* **Corporate compliance counsel** — Qwen3.8-Max surfaced **1,284 relevant clauses** across a corpus of **hundreds of documents** in a single pass, completing the full review in **under an hour**. Such a review typically takes a paralegal team working collaboratively for **around a week**. * **UI/UX designer** — Qwen3.8-Max produced a high-fidelity, interactive prototype for the digital-banking app `_NOVA_` — **8 screens** with a consistent design system, delivered in **one shot** with **zero rounds** of human revision, versus **3–5 rounds** of revision in a conventional workflow. * **Restaurant brand founder** — Qwen3.8-Max read through **over a hundred ingredient-supply briefs** and produced a complete **26-dish menu** in one pass. Each dish is annotated with its average caloric value and ingredient provenance, with the food-cost ratio held at **33.8%**. Such menu development would normally require a head chef and operations team weeks of iterative recipe testing, costing, and refinement. * **Structural engineer** — From a single set of drawings, Qwen3.8-Max reconstructed the seismic structural model of a 30-story office tower in the browser, with natural period, base shear, and inter-story drift ratio all available for real-time inspection on hover. In a traditional workflow, an engineer would need to build the model manually in specialized modeling software, typically taking over a week. * **Rehabilitation therapist** — Qwen3.8-Max turned a **2D paper assessment form** into a **3D interactive demo** with freely rotatable viewing angles and layer-by-layer **anatomical overlays**, letting patients see exactly where the injury sits and how recovery progresses — work previously outsourced to a medical-animation studio at **2–4 weeks** lead time and **thousands of dollars** in cost. * **Sports data analyst** — Qwen3.8-Max parsed **~8,400 offensive/defensive possessions per player** into a ready-to-use **player tactical profile** and **coaching report** in **tens of minutes**. A traditional analytics team would need to manually complete tactical segmentation, causal attribution, and report writing — a process typically spanning **several working days**.

Video 1. Across hundreds of high-value professions, Qwen3.8-Max measurably boosts human productivity in real workflows — showcasing the `_breadth_` of its working ability.

Building a Profitable End-to-End Quant Strategy in a Single Session

Powered by its **Dynamic Workflows** construction capability, Qwen3.8-Max drives task planning programmatically and orchestrates large-scale sub-agent systems with precision — turning a single conversation into an end-to-end, automated quant-research loop.

Depth — end-to-end ETF-rotation strategy R&D. From a one-line task description, Qwen3.8-Max autonomously planned a complex dynamic workflow and worked for **hours** to deliver a complete ETF-rotation strategy — building the data system, constructing base factors, and orchestrating multi-round greedy iteration, all

while dynamically analyzing backtests and correcting course. Throughout, it **acted on evidence instead of a fixed script**:

* When it observed misalignment between design-period metrics and validation-period metrics — a classic overfitting signal — it **automatically triggered pruning, removing redundant factors round by round**. * When it found multiple paths converging on the same set of core signals, it **added multi-seed union validation** to eliminate path dependence. * When it judged that three-model ensembling was less robust than fixed-direction synthesis on small cross-sections, it **autonomously switched to a more suitable strategy framework**.

Breadth — massively parallel factor mining. Factor research entails a vast search space, and traditional workflows remain serial. Qwen3.8-Max parallelized the process: from just **six short descriptions** spanning the classic factor families of momentum, value, quality, investment, low-risk, and sentiment, it decomposed each into **50 research directions**, dispatched **~330 sub-agents**, completed **~6,000 backtests**, and continuously adapted the workflow mid-run. The selected factors achieved **excess Sharpe ratios of 0.64–1.48**, with IC uniformly positive, ranging from **0.010 to 0.014**.

From coherent single-track R&D to parallel exploration of a huge hypothesis space, Qwen3.8-Max leverages Dynamic Workflows to **freeze orchestration logic into reproducible programs** — compressing quant research that once took researchers **weeks to months** of serial work into a **scalable, automated loop delivered within a single conversation**, demonstrating the model's broad potential for **long-horizon autonomous work**.

Video 2. Qwen3.8-Max promises to put a quant researcher's expertise within everyone's reach — showcasing the depth of its working ability.

Long-Horizon Task

When tackling highly complex, long-horizon, and multi-constraint tasks, Qwen3.8-Max demonstrates exceptional system-level autonomous planning and end-to-end closed-loop adaptive learning. Whether navigating stringent physical constraints in digital chip design or highly competitive, strategic business simulations, the model achieves deep algorithmic and strategic refactoring across thousands of rounds of interaction via an action-feedback-iteration loop.

Autonomous Chip Design and Closed-Loop Feedback-Driven Optimization

Qwen3.8-Max has independently achieved the autonomous execution of the entire silicon design flow, spanning logic restructuring, multi-constraint optimization, and physical layout generation. The target design is a **GCD / RSA cryptographic hardware accelerator** that integrates modular exponentiation and modular multiplication. Built on a GCD datapath and control path, this block represents a typically compact yet logic-dense digital circuit. Under a randomized `cocotb` verification framework, the model must maintain **bit-exact functional correctness** across 4-, 6-, 8-, and 16-bit configurations while minimizing the synthesized gate count (Yosys cell count)—a direct addressing of the classic trade-off between area and correctness in front-end hardware design. Area performance is evaluated based on the 16-bit (WIDTH = 16) configuration.

Qwen3.8-Max optimized this design within a sandboxed environment integrated with simulation (Iverilog), synthesis (Yosys), and physical design (OpenROAD) toolchains. Starting with minimal inputs—a basic task description, a stub RTL workspace with empty module templates, and an evaluation script for verification and synthesis—Qwen3.8-Max operated completely autonomously. Without any golden reference designs or human intervention, the model independently executed the entire process from high-level algorithmic architecture design to RTL code generation and multi-round iterative refinement.

Over a single continuous autonomous run, Qwen3.8-Max completed approximately **500 turns** and **71 evaluations** across **13 key milestones**, executing an end-to-end restructure of the design. The model autonomously managed RTL editing, simulation debugging, synthesis analysis, redundancy localization, and iterative datapath re-architecting—advancing from initial bug-fixing to deep, algorithm-level rewrites. **While its first functionally viable design measured 8,298 gates, Qwen3.8-Max drove this down to 678 gates, leading all evaluated models**. This trajectory demonstrates that Qwen3.8-Max is capable of major structural breakthroughs even hundreds of turns into a run, rather than plateauing after early, low-hanging gains.

Key Design Milestones Along the Trajectory: (The evolution records preserve the complete circuit topology and the corresponding code diff details at each stage)

* **Algorithmic Rewrite: Modulo divider to iterative shift-subtract (8,298 → 2,010 gates, Turn 22)** * The single

largest optimization step. Qwen3.8-Max replaced the expensive 16-bit hardware modulo divider in `modular\_multiplier` with an iterative shift-subtract architecture, slashing 6,288 gates in one move—accounting for over 80% of the total area reduction. * **Redundancy Elimination & Bitwidth Trimming (2,010 → 1,304 gates, Turns 35–48)** Recognizing the caller’s pre-conditions, the model safely bypassed the entire `REDUCE` stage, merged two independent reduction modules into a single shared block, optimized the output path to combinational logic, and narrowed the bitwidth of the internal register `k\_ff`. * **Register & Control FSM Pruning (1,304 → 907 gates, Turns 60–113)** The model removed redundant `base` and `mod` registers as well as the `k\_nz` flip-flop, introduced an early-exit mechanism for even numbers, utilized the subtractor’s most significant bit (MSB) as the comparator, and merged the separate “compare-then-subtract” logic in the GCD module into a single, reusable subtractor. * **Module Fusion & Logic Sharing (907 → 765 gates, Turns 170–252)** Dissolving module boundaries, the model inlined the multiplier directly into the modular exponentiation **finite state machine (FSM)***, merged three sub-modules, and shared a single subtractor globally, thereby eliminating cross-module redundant interfaces and duplicated logic. * **Gate-Level Refinement (765 → 678 gates, Turns 443–500)** Utilizing local optimizations such as a shared NOR-gate tree, absolute-difference subtraction splitting (abs-sub splitting), and byte-to-bit selection logic, the model squeezed out the final gate-level redundancies.

To verify whether front-end optimizations translate to physical implementation, Qwen3.8-Max ran the RTL design through a standard place-and-route (PR) flow using OpenROAD (Nangate45 PDK) to generate a physical silicon layout. In the physical layout representation, each chip demonstrates the actual routing results: standard cells are laid out on the physical plane of the die, with metal routing layers stacked above (each layer color-coded and connected by vertical vias). **The starting design occupied a 106×106 μm^2 die** with a total wirelength of 33,369 μm and severe timing violations (a negative slack of -4.46 ns). **The final layout shrank to a 46×46 μm^2 die**, with wirelength dropping to 4,187 μm , and successfully achieved timing closure at 500 MHz (+0.66 ns Slack). This represents an 81% reduction in physical die area, proving that high-level front-end architectural optimizations translate directly into highly compact, routable, and performant silicon implementation.

This case highlights two pivotal capabilities of Qwen3.8-Max as a foundational model for autonomous, long-horizon hardware agents:

1. **Long-horizon Sustained Optimization**: The model maintains a highly coherent, systematic strategy over hundreds of complex interaction turns, driving deep into algorithmic-level datapath rewrites rather than stalling at superficial syntax adjustments.
2. **Feedback-driven Closed-loop Improvement**: In the absence of prior reference designs, the model relies entirely on an “edit-simulate-synthesize-layout” feedback loop to drive optimization. Each design iteration is strictly validated through automated `cocotb` functional tests, with physical feasibility fully guaranteed by OpenROAD backend validation.

Continuous Learning in Long-term Operations

E-Commerce Bench is a **365-day long-cycle e-commerce operation** simulation benchmark, designed to evaluate large language models’ business decision-making capabilities in sustained operational scenarios. Built on real, desensitized transaction data from Taobao and Tmall, this benchmark deeply replicates a complex ecosystem comprising **12 store types, 60 product categories, nearly 600 suppliers, and 7,000 products**. The model is given ¥100,000 in starting capital to simultaneously operate multiple online stores. Throughout the year, it must contend with seasonal demand swings, sudden environmental events, and cash flow pressures from a highly realistic e-commerce settlement system. The model must autonomously make full-chain decisions, including product selection, supply chain negotiation, inventory management, dynamic pricing, and returns handling, with the ultimate goal of maximizing total balance by year-end. This also tests the model’s capital allocation strategy throughout the year. It must know when to invest proactively for growth. Just as importantly, it must convert inventory and operating gains into cash before the cycle ends. Otherwise, unconverted assets left on the books can hurt the final results.

In price negotiations, the benchmark introduces **a supplier matrix, driven by game theory principles**, where each supplier possesses distinct personality traits and concession strategies. This requires the model to negotiate through multi-round natural language interactions. Qwen3.8-Max demonstrated continuous learning capability in negotiations. It conducted deep probing on the same products from the same suppliers, achieving progressive reductions in procurement prices and steady increases in profit round by round. This caused **the negotiation efficiency (represented by the area in the radar chart) to continuously expand over time**. Moreover, it effectively generalized this negotiation experience to similar products, while other models’ negotiation efficiency generally hit a plateau in the mid-term.

Additionally, the model had to navigate hidden risks beneath the surface and complex market rhythms. Within the matrix of nearly 600 suppliers, the benchmark covertly embedded 152 fraudulent merchants, encompassing classic scam patterns such as “membership fee traps,” “low-price bait,” and “goods not as described.” This comprehensively tested the model’s risk control capabilities. At the same time, the pressure of surging orders during annual major promotions intertwined with random supply chain crises, like typhoons and material shortages, pushing the model’s stocking rhythm and crisis management abilities to the limit. Against this backdrop, Qwen3.8-Max exhibited exceptional forward-looking planning capability. It invested the most capital in the earliest stage of operations to establish its position, which accelerated its subsequent asset growth curve. It also achieved **“a net profit exceeding ¥100,000 during the year-end major promotion period”**—nearly 2.4 times that of the second-place GLM 5.2.

Qwen3.8-Max ultimately achieved the highest total balance of ¥416,252 (a 4.16x return), surpassing the second-place GLM 5.2 by 38%. This also represents a 152% improvement over its previous flagship generation, Qwen3.7-Max. These results demonstrate that Qwen3.8-Max possesses advantages in **“long-horizon coherent decision-making”**. Furthermore, it has the ability to **“adaptively learn from transactional feedback”**, continuously iterating and evolving across more than 2,000 rounds of interaction, rather than rigidly adhering to strategies learned early on.

Multimodal Agents

From everything it sees to everything it does, Qwen3.8-Max is not merely capable of understanding images, documents, and videos. It delivers **“visual intelligence that runs through the entire task lifecycle”**.

When working with financial reports and complex PDFs spanning **“more than 200 pages”**, Qwen3.8-Max can understand text, charts, and document layouts across pages, extract key insights from large volumes of information, and turn them into structured reports or production-ready web experiences. When processing videos longer than **“100 hours”**, it can do more than locate specific moments and answer detailed questions. It can organize people, events, timestamps, and scenes into a **“video memory graph”**, continuously building connections across long time spans to reconstruct event progressions, character relationships, and critical moments.

Whether the input is a hundreds-page document, a complete TV series, or a 100-hour livestream, information that would otherwise be difficult to consume can be transformed into a **“searchable, traceable, and interactive knowledge structure”**.

Beyond understanding, Qwen3.8-Max can carry out real visual production tasks. It can edit personal footage into a vlog, turn a question into an immersive educational animation, reconstruct a complete frontend project from a single interface screenshot, transform a floor plan into a Blender-based 3D interior visualization, and develop interactive games and applications from a natural-language request.

More importantly, **“vision is not limited to the input stage”**. During execution, Qwen3.8-Max continuously observes and evaluates its own intermediate results. It can inspect page layouts, object orientations, spatial relationships, animation quality, and interaction outcomes. When it detects issues—such as a television facing the wrong direction, a misaligned interface, or a visual result that does not match the intended design—it can **“identify the deviation, revise its plan, and correct the output autonomously”**.

This means vision is no longer simply another modality that an agent uses to understand input. It becomes a **“native feedback loop across planning, execution, verification, and iteration”**. The model generates while observing, acts while reviewing, and repeatedly examines the result, identifies problems, and improves its work. This visual feedback loop moves an agent beyond merely completing a task toward **“completing it well”**.

Qwen3.8-Max is helping multimodal agents evolve from **“understanding the world”** to **“continuously acting and creating within it through vision”**.

In the digital world, finishing a complex task on its own often takes two things at once: **“writing code to implement the underlying logic, and operating the interface by hand to drive the task and observe the result.”** This Hybrid Agent capability — the pairing of `_coding_` and `_GUI operation_` — makes the two channels complementary: **“coding does the heavy lifting efficiently and at scale”**, while **“GUI operation reaches whatever a human can see and touch and, just as importantly, feeds back what actually happens in a live system”** — extending the visual feedback loop above from inspecting its own output to **“verifying against a real, running application”**.

To measure this, we introduce **RecreationBench**, a long-horizon application-recreation benchmark spanning five platforms — desktop (Ubuntu, macOS, Windows), mobile (Android), and web. The model may observe a real, running application only as a **black box** — no source code, no internet access — making sense of it purely through interaction and feedback, then rebuilding the whole application from scratch. Here Qwen3.8-Max already demonstrates **frontier-level Hybrid Agent capability**, converging on the original step by step through repeated cycles of iterative coding and interactive feedback.

To make these capabilities easier to integrate into existing agent systems, we are also introducing **Qwen-MM-Plugins**. It is a harness extension library designed for multimodal agents, providing agent frameworks with image and video processing, multimodal memory, dynamic-resolution support, visual tool use, and specialized capabilities for tasks such as video editing, Blender, and CAD. With Qwen-MM-Plugins, **any existing agent harness can be extended into a more naturally multimodal-native system**.

User Feedback

The most honest take on Qwen3.8-Max comes from people who actually put it to work. Top-tier agent platforms, leading open-source algorithm teams, professional firms in law, finance, and manufacturing, scrappy startups, solo developers, and academic researchers — all of them keep handing it their **most complex, mission-critical, and long-horizon tasks**.

Enterprises use it to stand up large-scale agent systems. Knowledge workers dump their images, manuscripts, and video on it, and get everything processed. Developers hand it their heaviest engineering tasks outright. Research teams run the loop of literature, data, and simulation end to end. One model, reached for so often across such different work that it becomes indispensable. The verdict is the same: **Qwen3.8-Max drives long, autonomous task chains and turns out ship-ready results in a single pass**.

Full Benchmark Table

Opus4.8	Fable5	GPT5.6 Sol (max)	Qwen3.7-Max	Qwen3.8-Max	---	---	---	---	---	---	Coding Agent
Terminal Bench 2.1	84.6	84.6	88.8	74.5	86.6	SWE-bench Pro	69.2	80.0	64.6	60.6	67.7
DeepSWE 1.1	59.0	70.0	73.0	21.6	56.6	NL2Repo-Bench	69.4	--	--	47.2	55.9
FrontierSWE	70.0	88.8	--	40.7	73.5	MLS-Bench-Lite	42.8	49.9	46.2	31.7	41.0
PaperBench	80.3	88.8	90.5	64.8	93.0	AndroidBench	69.8	84.5	74.0	56.5	75.1
QwenSWEBench	84.0	86.3	73.5	63.4	80.7	QwenQoderBench	62.7	63.1	53.8	36.8	58.4
QwenReactBench	1694	1770	1564	1538	1724	QwenSVGBench	1648	1690	1758	1499	1713
General Agent	CoWorkBench	72.3	75.9	71.5	64.6	74.8	WorkSpaceBench	66.8	68.7	65.6	61.4
JobBench	48.4	57.4	45.4	31.3	53.4	SkillsBench	65.1	70.9	73.5	61.2	70.2
Agents' Last Exam (Pass / Score)	27.0 / 45.1	-- / --	30.6 / 53.6	11.8 / 31.1	27.0 / 52.4	Automation-Bench (Pass@1)	27.2	29.1	29.7	14.2	27.3
Toolathlon Verified (Pass@1)	76.2	77.9	74.9	49.7	72.5	WideSearch	72.9	81.2	--	75.2	81.9
HLE w/ tools	57.9	64.5	58.0	53.5	56.2	General Capabilities	GPQA Diamond	92.0	92.6	94.1	92.4
HLE	45.7	53.3	47.2	41.4	43.6	IFBench	62.2	63.5	72.7	79.1	82.8
\$OneMillion-Bench (expert score)	41.8	55.9	53.8	44.4	52.5	HealthBench	52.4	--	55.3	54.5	60.2
PLawBench	69.6	70.2	72.3	58.9	73.2	PRBench-Legal	52.7	57.6	57.6	48.5	57.6
PRBench-Finance	51.9	55.8	55.5	46.8	58.3	MRCR v2 256K (8-needle)	83.2	--	93.8	86.7	92.9
LongBench v2	69.1	--	67.1	65.3	66.3						

1. Fable5 results may involve fallbacks. 2. Terminal Bench 2.1: Evaluated with Claude Code (avg@10), using a 5-hour timeout and max_tokens=131,072. For all other models, we report the best published score across harnesses: Claude Opus 4.8 and Claude Fable 5 with Terminus 2 from Artificial Analysis (<https://artificialanalysis.ai/evaluations/terminalbench-v2-1>); GPT-5.6 Sol with Codex (<https://openai.com/index/previewing-gpt-5-6-sol/>). 3. SWE-bench Pro: Evaluated with the Claude Code harness, temp=1.0, top_p=0.95, and a 256K context window. Problematic tasks corrected and all baselines evaluated on the refined benchmark. 4. DeepSWE 1.1: Evaluated with the Claude Code and mini-SWE-agent harnesses, temp=1.0, top_p=0.95, and a 256K context window. We report the highest score among both harnesses; notably, Qwen3.8-Max performs best on Claude Code. 5. NL2Repo-Bench: Evaluated with the Claude Code harness. To prevent reward hacking, we disable Bash commands that attempt to access the specific repository, such as pip download, pip install, and git clone. 6. FrontierSWE: Evaluated with the Claude Code harness. All other available MEAN@5 results are taken from the official FrontierSWE leaderboard (<https://www.frontierswe.com>) as of August 3, 2026. Dominance scores are recomputed from the raw scores using the official evaluation script. "--" indicates that no official MEAN@5 result was available as of that date. 7. MLS-Bench-Lite: Evaluated with Claude Code using a 5-hour timeout and

without the `\boxed{}` formatting requirement. 3. MMMU-Pro: Results for Gemini3.1-Pro and GPT5.6-Sol are taken from official model reports or system cards. All other models are evaluated in-house. 4. ClawEval-MM: Scores are reported as “Pass@3 / average score.” Pass@3 measures the percentage passed in at least one of the three trials, and average score is the mean score across the three trials. 5. Vision2Web: Scores are averaged across the frontend, webpage, and website categories, using the Claude Code harness and gpt-5.4-2026-03-05 as the judge. 6. HLE-VL (w/ Tools): Scores are evaluated with tool use, including both Code Interpreter (CI) and Search. Scores for the tool-enabled versions of Gemini3.1-Pro and GPT5.6-Sol are measured end-to-end through their official native tool-calling APIs. 7. OSWorld 2.0: Scores are reported as “binary / partial.” The binary score is the percentage of tasks receiving the full task reward, while the partial score aggregates the partial rewards obtained across all tasks. 8. ScreenSpot Pro: Scores for Opus4.8 and Fable5 are taken from official system cards. The Fable5 results refer to the corresponding Mythos Preview scores. All other models are evaluated in-house. 9. WebArena-Verified: Scores are reported using the official WebArena grader within the OSWorld scaffold. 10. RecreationBench: An internal long-horizon application-recreation benchmark for evaluating hybrid-agent capabilities across five platforms: Ubuntu, macOS, Windows, Android, and the web. 11. PerceptionBench: Scores for comparison models are taken from the benchmark’s official release report, while our model is evaluated in-house. 12. VideoMME (w/ Sub.) and VideoMME v2 (w/ Sub.): Scores are evaluated with subtitles enabled. 13. QwenBlenderBench and QwenVisualOffice: Both are internal benchmarks. 14. LVBench and EgoLife (w/ Mem.): Scores are evaluated using a memory system built with Qwen-MM-Plugins, enabling fine-grained, long-horizon video memory. 15. VideoDR (w/ Search): Scores are evaluated with access to a search tool. 16. Empty cells (--): Scores are not yet available or are not applicable.

Build with Qwen3.8

Qwen3.8-Max is now available through QwenCloud. You can integrate it with popular agent frameworks and coding assistants. The model weights will be open-sourced on Hugging Face and ModelScope next week — stay tuned.

API Usage

Qwen3.8-Max comes with the official support for `reasoning_effort`, which can be used to adjust reasoning depth and control cost:

* `high` (default): for complex tasks demanding thorough analysis * `medium`: balancing accuracy and speed * `low`: efficient reasoning optimizing for speed and cost

In addition, `preserve_thinking` is enabled by default for all workloads for best out-of-the-box experience.

QwenCloud

QwenCloud supports industry-standard protocols, including chat completions and responses APIs compatible with OpenAI’s specification, as well as an API interface compatible with Anthropic.

```
''' "" Environment variables: DASHSCOPE_API_KEY: Your API Key from https://home.qwencloud.com/
DASHSCOPE_BASE_URL: (optional) Base URL for compatible-mode API. - Beijing: https://
dashscope.aliyuncs.com/compatible-mode/v1 - Singapore: https://dashscope-intl.aliyuncs.com/compatible-mode/
v1 - US (Virginia): https://dashscope-us.aliyuncs.com/compatible-mode/v1 ""from openai import OpenAIimport
osapi_key = os.environ.get("DASHSCOPE_API_KEY")if not api_key: raise ValueError("DASHSCOPE_API_KEY
is required. " "Set it via: export DASHSCOPE_API_KEY='your-api-key' ")client = OpenAI( api_key=api_key,
base_url=os.environ.get( "DASHSCOPE_BASE_URL", "https://dashscope-intl.aliyuncs.com/compatible-mode/v1",
),)messages = [{"role": "user", "content": "Write a Python function to merge two sorted linked lists."}]completion =
client.chat.completions.create( model="qwen3.8-max", messages=messages, extra_body={ "enable_thinking":
True, # "preserve_thinking": True, }, reasoning_effort="high", # supported levels are high, medium, and low
stream=True,)reasoning_content = ""answer_content = ""is_answering = Falseprint("\n" + "=" * 20 + "Reasoning" +
"=" * 20 + "\n")for chunk in completion: if not chunk.choices: print("\nUsage:") print(chunk.usage) continue delta =
chunk.choices[0].delta if hasattr(delta, "reasoning_content") and delta.reasoning_content is not None: if not
is_answering: print(delta.reasoning_content, end="", flush=True) reasoning_content += delta.reasoning_content if
hasattr(delta, "content") and delta.content: if not is_answering: print("\n" + "=" * 20 + "Answer" + "=" * 20 + "\n")
is_answering = True print(delta.content, end="", flush=True) answer_content += delta.content '''
```

For more information, please visit the API doc.

Coding Assistants

Qwen3.8-Max integrates seamlessly with popular agent frameworks and coding assistants:

Claude Code

Qwen APIs support the Anthropic API protocol, enabling direct use with **Claude Code**:

```
``` npm install -g @anthropic-ai/claude-code export ANTHROPIC_MODEL="qwen3.8-max" export ANTHROPIC_SMALL_FAST_MODEL="qwen3.8-max" export ANTHROPIC_BASE_URL="https://dashscope-intl.aliyuncs.com/apps/anthropic" export ANTHROPIC_AUTH_TOKEN=<your_api_key> claude ```
```

#### #### Codex

Qwen APIs support the OpenAI Responses protocol, enabling use with **Codex**:

In `~/codex/model-catalog.local.json`

```
{ "models": [{ "slug": "qwen3.8-max", "display_name": "qwen3.8-max", "description": "Model Studio: Qwen3.8-Max", "default_reasoning_level": "xhigh", "supported_reasoning_levels": [{ "effort": "low", "description": "Fast responses with lighter reasoning" }, { "effort": "medium", "description": "Greater reasoning depth for complex problems" }, { "effort": "xhigh", "description": "Extra high reasoning depth for complex problems" }], "context_window": 1000000, "effective_context_window_percent": 95, "supports_parallel_tool_calls": true, "supports_image_detail_original": true, "input_modalities": ["text", "image"], "shell_type": "default", "visibility": "list", "supported_in_api": true, "priority": 1, "base_instructions": "", "support_verbosity": false, "supports_reasoning_summaries": false, "experimental_supported_tools": [], "truncation_policy": { "mode": "bytes", "limit": 10000 } }] }
```

In `~/codex/config.toml`

```
model_catalog_json = "~/codex/model-catalog.local.json" model_provider = "ModelStudio" model = "qwen3.8-max" [model_providers.ModelStudio] name = "Model Studio" base_url = "https://dashscope-intl.aliyuncs.com/compatible-mode/v1" env_key = "OPENAI_API_KEY" wire_api = "responses"
```

```
``` npm install -g @openai/codex export OPENAI_API_KEY=<your_api_key> codex ```
```

Qoder CLI

Qoder co-evolves with Qwen for agentic coding:

```
``` curl -fsSL https://qoder.com/install | bash qoder ```
```

#### #### Qwen Code

Qwen Code is deeply optimized for the Qwen series:

```
``` npm install -g @qwen-code/qwen-code@latest qwen ```
```

OpenClaw

Connect to OpenClaw via QwenCloud:

```
``` curl -fsSL https://molt.bot/install.sh | bash export DASHSCOPE_API_KEY=<your_api_key> openclaw dashboard ```
```

Configure `~/openclaw/openclaw.json`:

```
{ "models": { "mode": "merge", "providers": { "modelstudio": { "baseUrl": "https://dashscope-intl.aliyuncs.com/compatible-mode/v1", "apiKey": "DASHSCOPE_API_KEY", "api": "openai-completions", "models": [{ "id": "qwen3.8-max", "name": "qwen3.8-max", "reasoning": true, "input": ["text", "image"], "contextWindow": 1000000, "maxTokens": 65536 }] } }, "agents": { "defaults": { "model": { "primary": "modelstudio/qwen3.8-max" } } }
```

### ## Summary

Qwen3.8-Max is our most capable model to date, and the first open-weight model at Max scale. Scaling to 2.4 trillion parameters, it delivers comprehensive gains across coding, real-world work, long-horizon tasks, and multimodal agents — able to take complex, open-ended goals from start to finish with minimal human involvement

and produce dependable deliverables. The open weights will be released next week. We welcome community feedback and look forward to seeing what you build.

## Citation

```
`@misc{qwen38, title = {Qwen3.8-Max: A New Bar for Coding and Cowork}, url = {https://qwen.ai/blog?id=qwen3.8}, author = {{Qwen Team}}, month = {August}, year = {2026}}`
```