MCP SERVER | NO CODE | CLOUD HOSTED

Cerebras Inference MCP for AI Agents

Run Massive LLM Batch Processing and Chat Completions

Cerebras Inference gives your AI agent access to the Cerebras Wafer-Scale Engine (WSE), delivering industry-leading speed for all large language model tasks. Use this MCP to generate chat responses, run massive batch processing jobs, and discover models at record speeds. It's built for data scientists and developers who need near-instantaneous LLM performance.

A+ Quality Score 98.33/100

llm-inference | wafer-scale | high-speed-ai | llama3 | batch-processing

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/cerebras-inference

SCAN TO VIEW

The connectivity layer between AI and the world's software.

Vinkius sits between AI and every application. All communication passes through Vinkius Cloud via the Model Context Protocol (MCP) — with governance, observability, and security at every layer.

Your AI Connections Run Through Vinkius Cloud

The world's largest managed MCP catalog

Vinkius is the connectivity layer where AI connects to the software your business already runs. We handle the hosting, the security, the credentials, the uptime — you get agents that actually do things.

We operate the world's largest managed MCP catalog. Major SaaS platforms, CRMs, databases, and cloud providers — running, monitored, production-ready. This MCP server is hosted and maintained by the Vinkius Cloud for AI Agents.

> *The agent doesn't manage credentials, doesn't manage uptime, doesn't manage security. Vinkius does.*
>
> — Architecture principle

Four Pillars of the Vinkius Runtime

01 — Security by design

Credentials stay encrypted at rest via AES-256. The AI agent never touches raw keys — they're injected into a sandboxed V8 isolate at runtime. Actions are logged, and connections have an emergency kill switch.

02 — Built on MCP Fusion

This MCP server was built with **MCP Fusion**, the open-source framework (Apache 2.0) that powers the entire Vinkius catalog. Schema-as-firewall strips undeclared fields, compiled PII redaction runs at zero overhead, and cryptographic lockfiles produce git-diffable audit trails.

03 — Deterministic observability

Eight immutable metrics per endpoint: request volume, p95 latency, error rate, active connections, cost attribution. A live payload feed logs every tool call with mutation detection.

04 — Autonomous operations

Servers are deployed, monitored, and patched autonomously. New capabilities and security patches ship weekly. Zero-downtime deployments ensure continuous availability across all managed MCP servers.

AES-256	Ed25519	24h TTL	V8 Isolate
Encryption at rest	PKI vault signatures	Ephemeral session keys	Sandboxed execution

One Token. Instant Access.

Every MCP server on Vinkius is accessed through a **Connection Token**. Tokens are generated in the cloud dashboard and produce a unique MCP endpoint URL. Paste this URL into any MCP-compatible client — no SDK required.

A single token can serve **multiple AI clients simultaneously**, or you can issue separate tokens per client for granular access control. Each token tracks its own request count, last activity timestamp, and can be individually enabled or revoked.

MCP ENDPOINT

```
https://edge.vinkius.com/{token}/mcp
```

Claude · Cursor · VS Code · Windsurf · Grok · Gemini

Security Is the Architecture

Security in Vinkius is not a feature — it's the foundation of the runtime. The gateway enforces multiple independent protection layers between AI agents and third-party APIs.

01 — Ed25519 PKI Vault

Every workspace has an Ed25519 Master Key. Session keys are generated ephemerally (24h TTL) and signed by the Master Key. Credentials never leave the vault boundary.

02 — V8 Isolate Sandboxing

Tool code runs inside isolated-vm V8 isolates with 64 MB memory caps and per-request timeouts. No filesystem access, no network access except through the SSRF-guarded fetch bridge.

03 — SSRF Guard

All outbound HTTP requests are DNS-resolved and validated before execution. Private IP ranges (10.x, 172.16-31.x, 192.168.x, AWS metadata 169.254.x) are blocked at the network layer.

04 — DLP & PII Redaction

A ResponseGuard pipeline intercepts every tool response. Configurable redaction patterns strip sensitive fields (emails, SSNs, card numbers) before data reaches the AI agent.

05 — Cryptographic Audit Trail

Every request is signed into a SHA-256 hash chain with Ed25519 signatures. Events form a tamper-proof, SIEM-exportable forensic record.

06 — Honeytoken Trap System

Phantom credentials are injected into isolated environments. If a honeytoken is used outside Vinkius infrastructure, the server is quarantined instantly.

Emergency Kill Switch

EU AI Act Art. 14(1) Compliant

The kill switch is an **emergency halt** mechanism — not a simple toggle. When triggered, it executes three actions atomically:

01 — Server deactivated

The MCP server is immediately taken offline across the entire cluster.

02 — All tokens revoked

Every connection token is invalidated. Total lockout — reconnection blocked until new tokens are issued.

03 — WebSocket connections killed

Active connections terminated via Redis pubsub broadcast. Propagates to every runtime node in the cluster.

Full Visibility. Zero Guesswork.

The Vinkius cloud dashboard includes a full MCP Governance suite — real-time analytics and security controls for production AI operations.

Control Plane

KPI dashboard with request volume, latency, success rate, token consumption, and AI-generated operational briefings.

Agent Activity

Which AI clients are connecting, how often, and what they're doing — real-time session tracking.

FinOps

Cost tracking per tool, payload compression savings, budget optimization signals, and consumption trends.

Tool Health

Slowest and most error-prone tools, with actionable root-cause insights and performance baselines.

Firewall & DLP

PII redaction activity, sensitive data protection counters, and security event timeline.

Incident Log

Error trends, failure rates, status-code breakdowns, and forensic audit trail access.

> Get started at **cloud.vinkius.com** — connect your AI agent in under 60 seconds.

Cerebras Inference MCP

15 tools available
Cloud-hosted on Vinkius

Working with huge language models often means waiting forever for a response or struggling to process large datasets sequentially. This MCP changes that entirely. You can connect your agent through Vinkius, giving it access to the Cerebras Wafer-Scale Engine (WSE). What this means in practice is speed at scale. Your agent doesn't just generate chat completions; it does so with a massive boost of processing power. Need to run thousands of prompts against a dataset? You can queue those jobs for asynchronous batch processing, letting your workflow continue while the heavy lifting happens in the background. It’s ideal whether you need quick conversational responses or complex, multi-step data pipelines. When latency is critical—whether for product integration or research—this connection delivers the horsepower needed to keep up with modern AI demands.

Core Capabilities

01 — Generate Conversational Responses

The agent generates structured, high-speed chat completions suitable for dialogue flows.

02 — Process Large Datasets in Batches

You set up large workloads to run asynchronously and retrieve the results when they're ready, perfect for massive data processing.

03 — Manage Inference Files

The agent can upload JSONL files needed for batch jobs and download raw content once the process is complete.

04 — Discover and Inspect Models

You list available models or fetch detailed information to ensure you're using the right engine for your task.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/cerebras-inference** — connect your AI agent in three steps.

01 First, subscribe to this MCP and input your Cerebras API Key into your AI client.

02 Next, instruct your agent on the required action—for example, queuing a batch job or generating a chat completion using a specific model.

03 Finally, the engine executes the task at high speed, returning structured results, status updates, or downloadable files to your agent.

The bottom line is that you get extremely fast access to advanced LLM processing without worrying about underlying hardware limitations.

Built For

This MCP targets data science teams, ML developers, and product engineers whose core workflow relies on running large language model inference at scale. If your job involves analyzing massive datasets or building consumer-facing AI features with low latency requirements, this is for you.

Machine Learning Engineer

They use the MCP to build and test applications requiring near-instantaneous model responses, maintaining development momentum while integrating complex models.

Data Scientist

They leverage the asynchronous batch API to run large-scale inference across massive datasets without needing manual intervention or waiting for sequential processing.

Product AI Lead

They integrate high-performance LLMs into production environments where any significant latency factor could degrade user experience.

What Changes When You Connect

01 You get instant conversational responses using `create_chat_completion` and `create_completion`, eliminating chat latency issues.

02 Manage huge datasets with asynchronous jobs. Use `create_batch` to queue work, and then check status later with `get_batch`. This keeps your agent flow smooth.

03 Keep track of all your data pipelines by listing all runs using `list_batches` or viewing what files are uploaded via `list_files`.

04 When you need model details before running a job, use `list_models` to see every supported engine and check which ones match your task requirements.

05 Monitor performance directly. Call `get_metrics` to gather Prometheus-formatted data on your usage, helping you optimize costs.

Real-World Applications

Analyzing Customer Feedback at Scale

Instead of running a single prompt against 100 customer reviews manually, the agent uses `create_batch` to submit all JSONL files. It processes thousands of records overnight and then retrieves the summarized results using file tools.

Building Real-Time Chatbots

A developer needs a chatbot that feels natural, not robotic. Using `create_chat_completion` ensures the agent handles multi-turn dialogue correctly, making the user experience feel instantaneous.

Model Comparison for New Features

Before committing to a model choice, the Product Lead uses `list_models` and then `get_model` to fetch specific details, ensuring they select the engine that meets both speed and accuracy criteria.

Cleaning Up Old Jobs

A data science project ran a massive batch job by mistake. The engineer quickly uses `list_batches` to find the rogue ID and then calls `cancel_batch` to stop the unnecessary processing immediately.

Patterns to Avoid

Trying to process everything in one call

✗ AVOID

Asking your agent to run a chat completion, upload 5 files, and list 10 models all within a single prompt. This overwhelms the request and fails.

✓ INSTEAD

Break it down: Use `list_models` first to pick an engine. Then use `upload_file` for data prep. Finally, use `create_batch` or `create_chat_completion` separately.

Ignoring job status checks

✗ AVOID

Creating a batch job with `create_batch` and then assuming the results are ready immediately without checking.

✓ INSTEAD

After creating the job, always follow up by calling `get_batch` until the status is marked 'completed'. Once done, you can download the data using file tools.

Using deprecated model names

✗ AVOID

Attempting to run an inference with a model name that has been retired or isn't available for the current job type.

✓ INSTEAD

Always start by running `list_models` to guarantee you are targeting a currently supported engine, then use `get_model` if you need specific details.

The Right Fit

Use this MCP if your primary bottleneck is LLM inference speed or processing large volumes of data. You *must* use it when running batch operations, as the asynchronous tools like `create_batch`, `list_batches`, and `get_batch` are built for that scale. However, don't use this if all you need is simple text generation from a single prompt; while `create_completion` works, remember its primary strength is high-throughput batch processing. If your workflow requires complex external API calls outside of LLM inference (like interacting with a CRM or database), then look at other types of MCPs for those specific integrations.

Cerebras Inference MCP for AI Agents: High-Speed Model Batch Processing

Today, running large models often means hitting a wall. You have massive datasets or thousands of user inputs that need processing, but your current setup forces you to process them one after the other—a tedious cycle of API calls and waiting for responses.

With this MCP, that manual queueing disappears. Your agent uses the dedicated batch tools to send hundreds of files at once. You initiate the job and walk away; when it's done, the results are ready for you to download, allowing your workflow to move instantly from input preparation to final output consumption.

Cerebras Inference MCP for AI Agents: Model Discovery and Chat Dialogue

Before running any job, developers waste time checking model availability across different documentation tabs or writing boilerplate code just to list supported engines. This adds friction and risk of using the wrong configuration.

Now, your agent handles discovery automatically. It can run `list_models` to show you all options available, then use `get_model` to give you specs on a specific engine—all in plain conversation. You just get the right model, fast.

Cerebras Inference: 15 Tools for LLM Data Batch Processing

Use these tools to list models, create batch jobs, upload files, monitor job status, and get model metrics directly from your agent.

#	TOOL	DESCRIPTION
01	`cancel_batch`	Stops a batch job that is currently running or queued.
02	`upload_file`	Sends and uploads a JSONL file required for processing in a batch job.
03	`create_chat_completion`	Generates responses formatted for structured, back-and-forth conversational dialogue.
04	`create_completion`	Outputs continuations of text based on a single input prompt string.
05	`create_batch`	Initiates a large-scale, asynchronous job to process many inputs at once.
06	`delete_file`	Removes an uploaded file from the system storage.
07	`get_batch`	Checks and retrieves the current status and details of a specific batch job.
08	`get_file_content`	Downloads the raw text or data content from an uploaded file.
09	`get_file`	Retrieves metadata, such as size and owner, for a specific stored file.
10	`get_metrics`	Fetches operational usage data in Prometheus format for performance monitoring.
11	`get_model`	Retrieves detailed information about a specific model available on the platform.
12	`list_batches`	Lists all batch jobs that have been created or are currently pending.
13	`list_files`	Shows a list of all files previously uploaded for processing.
14	`list_models`	Retrieves a comprehensive list of every model currently supported by the system.
15	`list_public_models`	Lists models that do not require an API key to be viewed or selected.

See It in Action

Real prompts you can use once this MCP is connected to your AI agent through Vinkius Cloud.

U What are the fastest models I can use for chat completions?

Available High-Speed Models

MODEL NAME	CONTEXT WINDOW	MAX TOKENS/SEC
llama3.1-70b	8K	120
gpt-oss-120b	32K	95

Tip: For general dialogue, `llama3.1-70b` is your best bet for low latency.

U I have 5,000 reviews. How do I process them all at once?

You need to run this as a batch job.

Process Steps:
1. Upload your JSONL file first (e.g., `upload_file`).
2. Then, initiate the job (`create_batch`).
3. Use `get_batch` periodically to monitor progress and ensure it finishes successfully. Once done, download the results using the file content tool.

U List all available models and their capabilities.

Here are the primary model families:

- **Llama 3.1:** Great for conversational flow, high speed.
- **GPT-OSS-120b:** Best if you need large context windows for deep analysis.

Which one should we check out first? I can run `get_model` to give you the full specs on any of these.

Frequently Asked Questions

01 **How does Cerebras Inference MCP handle processing huge datasets?**

It uses an asynchronous batch API. You upload your data, queue the job, and then check back later for results. This means you don't wait through hours of processing time; your agent just checks when it’s ready.

02 **Is Cerebras Inference MCP better than other LLM APIs for chat?**

The strength here is the speed and reliability of the underlying engine. It provides consistently low latency across conversational turns, which makes your application feel much more responsive to the user.

03 **Can I use Cerebras Inference MCP if my model isn’t Llama 3?**

No problem. The platform supports multiple state-of-the-art models. You can use the listing tools within the MCP to discover and select exactly which engine you need for your specific task.

04 **What if my batch job fails? Can I fix it?**

Yes, you can monitor the job status using `get_batch`. If something goes wrong, you can sometimes cancel and restart the process or review the error logs to pinpoint where the failure occurred.

05 **Does Cerebras Inference MCP help with cost optimization?**

It helps by allowing efficient resource management. You can use the monitoring tools in the MCP to track your usage and optimize your inference workflows, making sure you're not paying for unused compute time.

06 **How do I get model details using Cerebras Inference MCP?**

You simply ask the agent to fetch the model information. The MCP will use `get_model` to retrieve detailed specs, letting you know about context limits and performance before you commit to a job.

Go Live in 60 Seconds

Get your connection token from **cloud.vinkius.com**, then paste the endpoint URL into any MCP-compatible client.

YOUR MCP ENDPOINT

```
https://edge.vinkius.com/[TOKEN]/mcp
```

CLIENT	WHERE TO CONFIGURE
Claude AI	Profile → Customize → Connectors → "+" → Add custom connector → Paste endpoint
Cursor	Settings → Features → MCP Servers → "+ Add New MCP Server" → Type: SSE → Paste endpoint
VS Code	Ctrl/Cmd+Shift+P → "MCP: Add Server" → add `"cerebras-inference": { "url": "…" }`
Windsurf	MCP Settings → `mcp_settings.json` → Add endpoint URL
ChatGPT	Settings → Tools & plugins → Add MCP server → Paste endpoint
Gemini	Extensions → Add MCP Server → Paste endpoint URL

ASK AN AI ABOUT THIS

Let your preferred AI explain this MCP server

- Ask ChatGPT
- Ask Claude
- Ask Perplexity
- Ask Gemini
- Ask Grok

READY TO CONNECT

Cerebras Inference is live on Vinkius Cloud.

Get your connection token, paste it into your AI agent, and start building. No SDK. No deployment. Just results.

Start at cloud.vinkius.com →

vinkius.com · support@vinkius.com

INDEPENDENT PLATFORM DISCLAIMER

Vinkius is an independent platform and is not affiliated with, endorsed by, sponsored by, verified by, or otherwise authorized by Cerebras Inference. All third-party trademarks, logos, and brand names are the property of their respective owners. Their use in this document is strictly for informational purposes to identify service compatibility and interoperability.

DOCUMENT INFORMATION

Generated	June 2026
MCP Server	Cerebras Inference MCP
Server ID	`019e3875-f162-719b-aa09-bc030c2f119c`
Platform	Vinkius Cloud for AI Agents
Endpoint	`https://edge.vinkius.com/{token}/mcp`

LICENSE & USAGE

This document is generated automatically by the Vinkius PDF Engine. Content reflects the MCP server configuration at the time of generation and may change as updates are deployed. For the most current information, visit vinkius.com/mcp/cerebras-inference.