MCP SERVER | NO CODE | CLOUD HOSTED

Hugging Face MCP

Research Models, Datasets, and Spaces Instantly.

Hugging Face MCP connects your AI agent directly to the world's largest hub for machine learning resources. Use it to find, inspect, and manage thousands of models, datasets, and live demo apps in one conversation. You can search by task type, review model file structures without downloading anything, track community discussions, or list available datasets—all from your preferred AI client.

A+ Quality Score 98.33/100

model-discovery | machine-learning | datasets | model-metadata | ai-research | pipeline-tasks

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

SCAN TO VIEW

The infrastructure that powers AI agents in the real world.

Vinkius connects AI to the world's software through secure, enterprise-grade infrastructure — enabling real-world execution at scale, built on the Model Context Protocol (MCP).

Your AI Connections Run Through Vinkius Cloud

The world's largest managed MCP catalog

Vinkius is the cloud infrastructure where AI agents connect 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.

Hugging Face MCP

13 tools available
Cloud-hosted on Vinkius

Connecting to the Hugging Face hub means you can treat your ML research like a natural conversation. Instead of switching tabs and manually searching across separate websites, your agent acts as an embedded data scientist. You can ask it to find models that perform specific tasks or locate datasets matching certain criteria. The tool lets you inspect model metadata—seeing tags, download counts, and file structures—all before deciding what's useful for your project. Need to check the status of a live demo app? Just ask. If you're working with Vinkius, this MCP makes sure that entire ecosystem of ML resources is accessible from a single point of entry, letting your AI client handle the heavy lifting. You can even use it to create discussions or browse existing community reports, keeping all your research notes right where they belong.

Core Capabilities

01 — Search and find models

Discover thousands of available ML models by filtering them using name, task type, framework, or author.

02 — Inspect model files

View the exact filenames, sizes, and paths within a model repository without having to download any weights or artifacts.

03 — Review datasets

List available datasets on the hub and view their descriptions, size details, and file trees for inspection.

04 — Check live demos (Spaces)

Retrieve information about ML demo applications, including whether they are currently running or down.

05 — Manage discussions

Read existing community threads for bug reports or feature requests, and also create new discussion topics on a specific model or dataset page.

One Click on Vinkius — From Prompt to Execution

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

01 First, subscribe to this MCP and provide your Hugging Face Access Token.

02 Next, tell your AI client what you're looking for—for instance, 'show me all image classification models using PyTorch.'

03 Finally, your agent returns the relevant model metadata, file lists, or discussion threads directly in the chat.

The bottom line is that you get an AI assistant capable of acting like a dedicated ML researcher, pulling data from Hugging Face without leaving your current interface.

Built For

This MCP is for the machine learning engineer who needs to vet dozens of model candidates quickly before committing code. It's for researchers drowning in datasets and developers needing to check live demo app status.

ML Engineer

They use this MCP daily to list models by task type, inspect file structures (using list_model_files) to ensure the right weights are present, and review community discussions before integration.

Data Scientist / Researcher

They browse datasets using list_datasets or look through model collections with get_collection to discover related resources for a new project without leaving their notebook environment.

ML Developer

They check Space runtime status with get_space and use the MCP to gather required model metadata (get_model_tags) needed for client-side application setup.

What Changes When You Connect

01	Stop leaving your AI client to check model tags or browse discussions. The MCP lets you get the full pipeline info (get_model_tags) and read community feedback (list_model_discussions) without opening a browser.
02	When vetting models, don't guess what files are inside. Use list_model_files to map out every single artifact—like config.json or model weights—before you download anything.
03	Need to find a good starting point? You can use list_models and filter by task type, quickly narrowing down thousands of options to only those relevant for your current project.
04	The MCP centralizes discovery. Whether you're listing datasets (list_datasets) or checking out live demo apps (get_space), everything is accessible from one conversation stream.
05	You can initiate conversations about models using create_discussion, making it easy to track bug reports and feature requests directly through your AI agent.

Real-World Applications

Vetting a new NLP model for production

A developer needs to know if a candidate model supports the right framework. They ask their agent to get_model_tags for several candidates, immediately seeing which ones are PyTorch and which are TensorFlow before writing any integration code.

Comparing dataset structures

A researcher needs two datasets but isn't sure how the files are stored. They use list_dataset_files on both, letting them compare file paths (e.g., 'train.parquet' vs 'data/raw') in a single view.

Checking if a demo app is operational

A team lead wants to know if the internal ML dashboard is working before a meeting. They ask get_space, and the agent instantly reports the current runtime status of the application.

Documenting a research finding

After testing several models, an engineer uses list_model_discussions to gather context on common bugs or optimal usage tips that others have already shared in the community threads.

Patterns to Avoid

Searching only by name

✗ AVOID

A user just searches for 'image classification' and gets a massive, unmanageable list of models with no useful filtering.

✓ INSTEAD

Don’t rely on simple text search. Use the explicit tools like list_models to filter results specifically by task tag or framework type, giving you a targeted set of candidates.

Assuming file contents

✗ AVOID

A developer assumes a model has weights in both PyTorch and TensorFlow formats because it's a popular model.

✓ INSTEAD

Always use list_model_files to confirm the exact file structure and available formats, ensuring you don't waste time trying to load non-existent artifacts.

Ignoring community context

✗ AVOID

A team implements a complex model without knowing about known memory leaks or optimal quantization techniques.

✓ INSTEAD

Before integration, always check list_model_discussions. The community threads are where you find the real-world usage tips and bug reports.

The Right Fit

Use this MCP if your core task is discovery, research, or metadata gathering around ML assets. You need to know *what* models exist, *how* they're structured (file system), and *if* the community has flagged issues. Don't use it if you just need raw data ingestion—for that, a direct API endpoint tool might be better. If your goal is simply comparing model performance against a specific benchmark metric, relying on existing scoring sheets outside of this MCP will be faster. But if your process involves vetting multiple candidates, inspecting metadata (get_model_tags), and understanding the file architecture, this MCP is essential.

The ML research workflow feels like constant context switching.

Today, finding a model requires a messy dance: you check Model A's card on one tab for its tags. Then, you open another tab to see if the dataset it uses is available and download its file list. Next, you jump over to a third tab just to read community discussions about potential bugs before writing any code.

With this MCP, your agent handles all that clicking. You ask one question—like 'Find me PyTorch models for image classification with good documentation'—and it pulls the tags, file structure details, and even related discussion links into a single conversation thread.

Inspect Model Artifacts Directly with list_model_files

The biggest manual headache is confirming exactly what weights or configuration files are inside a repository. You usually have to click 'Download' and then manually unzip the contents just to see if you got everything you needed.

Now, with list_model_files, your agent shows you every file name, size, and path instantly in plain text. It’s immediate validation of the entire model artifact inventory.

Hugging Face with 13 Tools

Use these tools to manage the full lifecycle of ML assets by listing files, creating discussions, retrieving metadata for models, and cataloging available datasets.

#	TOOL	DESCRIPTION
01	list_dataset_files	Lists all filenames within a specific Hugging Face dataset repository, helping you map out its structure.
02	create_discussion	Allows you to open a brand new conversation thread on any model, dataset, or space page.
03	get_collection	Retrieves specific details and information for a named Hugging Face collection slug.
04	get_model	Fetches core metadata about any specified model ID in the 'author/name' format.
05	get_model_tags	Provides detailed tags and pipeline information for a model, showing its framework, license, and primary task type.
06	get_space	Retrieves all operational details about a specific Hugging Face demo application (Space).
07	list_collections	Lists multiple curated model, dataset, and space collections available on the Hub.
08	list_datasets	Provides a list of datasets, along with their author, download counts, and creation dates.
09	list_model_discussions	Lists active discussion threads on a model page so you can review community feedback or bug reports.
10	list_model_files	Shows the full file list, sizes, and paths for any specified model repository directory.
11	list_models	Searches and returns a list of models on the Hub based on search terms or authors.
12	list_spaces	Lists available demo applications (Spaces), showing their SDK, title, and author.
13	get_user	Checks your authenticated user account details to confirm the token is working correctly.

See It in Action

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

U Find popular text generation models with over 1000 likes.

Here are top text-generation models by likes: meta-llama/Llama-3.1-70B (45k likes), mistralai/Mixtral-8×7B (12k likes), microsoft/Phi-3-mini (8k likes) and google/gemma-2-9b (6k likes). Would you like the file structure for any of these?

U Show me what files are in the bert-base-uncased model.

The bert-base-uncased repo contains: config.json (443 bytes), pytorch_model.bin (440 MB), tf_model.h5 (440 MB), flax_model.msgpack (440 MB), tokenizer.json (466 KB), tokenizer_config.json (48 bytes), vocab.txt (232 KB) and README.md. It has weights in PyTorch, TensorFlow and Flax formats.

U What discussions are happening on the Llama-3 model page?

There are 23 active discussions on meta-llama/Llama-3-8B. Top threads include: 'Fine-tuning with PEFT/LoRA — memory requirements' (18 replies), 'Quantization to 4-bit — GGUF format' (14 replies) and 'Comparison with Mistral-7B on reasoning tasks' (9 replies).

Frequently Asked Questions

01 **How do I use Hugging Face MCP to find all available models?**

You can list general candidates using list_models, which lets you filter by search term or author. It returns the model ID, task tag, and download count for quick comparisons.

02 **Can I use Hugging Face MCP to check a dataset's file structure?**

Yes, run list_dataset_files on your desired dataset repository. This gives you a clear list of every filename, like 'train.parquet', helping you understand the data layout.

03 **What is the best way to check model tags using Hugging Face MCP?**

Use get_model_tags and provide the full author/name ID for the model. This tool gives detailed information on its framework, license, and primary task tag in one go.

04 **How do I find live demo apps with Hugging Face MCP?**

Run list_spaces to get a catalog of all available demo applications. You can then use get_space on a specific ID to confirm its current runtime status.

05 **Can I create discussions using the Hugging Face MCP?**

Yes, you can initiate conversations with create_discussion. Just provide the repo type (model, dataset or space), the ID, and your title to start a new thread.

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 `"hugging-face": { "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

Hugging Face 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 Hugging Face. 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	Hugging Face MCP
Server ID	`019d8446-f4c1-71a2-997f-a18c4485c0fa`
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/hugging-face.