MCP SERVER | NO CODE | CLOUD HOSTED

YAML Parser Engine MCP

Reliable conversion for complex config files.

YAML Parser Engine converts data between YAML and JSON with absolute precision, even when dealing with complex configuration files that break other AI agents. It correctly handles advanced structures like anchors, aliases, and multi-document manifests. This is essential for safely processing Kubernetes, Docker Compose, or GitHub Actions configurations where a single misplaced space can cause the entire system to fail.

A+ Quality Score 100/100

yaml-parsing | serialization | kubernetes-config | data-validation | anchors-aliases

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/yaml-parser-engine

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.

YAML Parser Engine MCP

1 tools available
Cloud-hosted on Vinkius

Trying to run an AI agent on configuration files—especially YAML—is rough. These formats are brittle; they treat indentation like actual data and break silently when an anchor reference is dropped or a colon vanishes. This MCP uses a proven, industry-standard library to read and write these complex configs without losing any structural information. You feed it the content, tell it if you need JSON or YAML, and it spits out perfectly formatted code. If your AI agent needs to modify a Kubernetes manifest, or validate an Ansible playbook, this tool ensures that every field type, comment, and nested structure remains intact during the conversion process. Connecting this MCP via Vinkius gives your agent access to professional-grade data integrity, making complex infrastructure tasks reliably possible.

Core Capabilities

01 — Convert Kubernetes manifests

Change entire container or deployment configuration files between YAML and JSON.

02 — Validate CI/CD workflows

Process GitHub Actions workflow definitions to ensure they convert reliably for programmatic checks.

03 — Handle advanced data structures

Preserve complex YAML features like anchors, aliases, and merge keys during conversion.

04 — Process multi-document files

Read and convert configuration files containing several separate documents separated by '---'.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/yaml-parser-engine** — connect your AI agent in three steps.

01 You provide the MCP with a block of content and specify whether you want to convert it from YAML to JSON, or JSON back into YAML.

02 The engine processes the input using its robust parser, ensuring that all structural elements, including comments and advanced references, are correctly identified.

03 Your agent receives the fully converted output in the target format (JSON or YAML), ready for further processing.

The bottom line is you get perfectly formatted, structurally sound config data regardless of how messy the source file was.

Built For

This MCP is critical for anyone who works with infrastructure code or complex system definitions. It helps platform engineers and DevOps specialists stop debugging YAML indentation errors at 3 AM.

DevOps Engineer

Needs to programmatically validate that environment variables and service mappings defined in Docker Compose files are correct before deployment.

Site Reliability Engineer (SRE)

Must convert large, multi-document Kubernetes manifests into a structured format for automated policy checking or auditing.

Platform Developer

Uses the tool to test and validate complex CI/CD pipeline definitions, like GitHub Actions workflows, across different systems.

What Changes When You Connect

01 You eliminate structural failure risk. When converting Kubernetes manifests, you're guaranteed that fields like `spec.replicas` retain their correct data type and structure in the target format.

02 No more indentation errors stopping your build. The `parse_yaml` tool handles complex YAML 1.1/1.2 specs, ensuring accurate conversion for Docker Compose volume mappings.

03 Process files containing multiple documents (e.g., several separate service manifests) seamlessly. This MCP parses the entire file block and gives you all separated outputs.

04 The agent won't hallucinate data structure. Because the underlying library passes official YAML tests, the output is validated against real specs, not just guesswork.

05 Speed up CI/CD validation. You can reliably convert GitHub Actions workflows into JSON so your agent can validate conditional expressions and job dependencies before a merge.

Real-World Applications

Auditing Kubernetes deployments

An SRE needs to check 50 deployment manifests for consistency across replica counts. They feed the files into your agent, which uses `parse_yaml` to convert each YAML file to JSON. This allows the agent to run a consistent script against every single resource definition programmatically.

Debugging Docker Compose mappings

A developer is having trouble with volume definitions in `docker-compose.yml`. They use your MCP to convert the problematic YAML section to JSON, which immediately reveals if the colon mapping or service name structure was lost during an earlier manual copy/paste step.

Validating CI pipelines

The team needs to confirm that a new GitHub Actions workflow definition is valid before merging it into production. They use your MCP's `parse_yaml` tool to convert the entire workflow YAML to JSON, allowing automated testing of job dependencies.

Handling mixed config files

A project uses a single file containing both service definitions and related resource manifests. The agent feeds this multi-document YAML into your MCP, which correctly separates and converts all distinct configuration blocks to JSON for separate processing.

Patterns to Avoid

Using simple string replacement

✗ AVOID

Trying to convert a config file by just replacing 'key: value' with quotes, which fails when dealing with complex multi-line strings or special characters.

✓ INSTEAD

Use the `parse_yaml` tool. It is built specifically for YAML structure and handles all necessary escaping and type conversions automatically.

Relying on generic JSON parsers

✗ AVOID

Passing a deeply nested Kubernetes manifest to a standard JSON parser that chokes on anchors or aliases, resulting in incomplete data.

✓ INSTEAD

Use this MCP. It uses specialized libraries that understand the full YAML specification (1.1/1.2), ensuring zero structural loss when converting between formats.

Manually fixing indentation errors

✗ AVOID

A user spends an hour manually adjusting tabs and spaces in a Docker Compose file, only to discover the underlying service name is still wrong.

✓ INSTEAD

Use `parse_yaml` to convert the YAML to JSON first. This forces structural validation, making any human-induced indentation mistakes obvious before you proceed.

The Right Fit

You should use this MCP if your goal is converting complex configuration data (like Kubernetes manifests or CI/CD files) between YAML and JSON formats while maintaining absolute fidelity to the original structure. The key requirement is that the source file uses advanced YAML features like anchors, aliases, or multi-document separation.

Don't use this if you just need simple text manipulation—say, replacing every instance of 'foo' with 'bar'. For those tasks, a basic string utility tool works fine. If your data structure is guaranteed to be simple key/value pairs without any advanced YAML features, other general-purpose parsers might suffice. But when dealing with actual infrastructure code that relies on the full YAML spec, this MCP is required.

The headache of configuration files in AI agents

Right now, if your agent needs to read a complex setup file—say, defining networking rules for Kubernetes—you're usually dealing with copy-pasting and tedious manual validation. You have to worry about indentation being off by one space, or an anchor reference breaking because the AI misinterpreted the context.

With this MCP, you hand the raw configuration data directly to your agent. It uses robust parsing logic that understands YAML as a structure, not just text. What you get back is perfectly converted JSON or YAML that your agent can trust and act on immediately.

Using the `parse_yaml` tool delivers reliable data structures

Manual validation requires opening multiple tabs: one for the source, one for the JSON output, and a third to cross-reference missing fields. You spend time hunting down if a volume mapping or a service name was lost during conversion.

The `parse_yaml` tool handles all that complexity internally. It ensures bidirectional fidelity—meaning it converts YAML to JSON and back again without losing any data integrity. Your agent gets the clean, validated result every single time.

YAML Parser Engine: 1 Tool Available

Use this single tool to convert configuration content between YAML and JSON formats for structured data processing.

#	TOOL	DESCRIPTION
01	parse_yaml	Converts configuration content between YAML and JSON formats, supporting advanced features for Kubernetes and Docker Compose configs.

See It in Action

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

U Convert this Kubernetes deployment manifest to JSON so I can programmatically modify the replica count.

JSON output with spec.replicas, containers, volumes — all fields preserved with correct types.

U Our CI team needs the GitHub Actions workflow as JSON to validate it programmatically before merge.

Workflow parsed: jobs, steps, environment variables, and conditional expressions all converted to typed JSON.

U Take this Docker Compose JSON config and generate valid YAML for the docker-compose.yml file.

Valid docker-compose.yml generated with correct indentation, service definitions, and volume mappings.

Frequently Asked Questions

01 **Does the YAML Parser Engine MCP handle Kubernetes manifests?**

Yes, absolutely. This MCP is designed for infrastructure configs and handles complex fields found in Kubernetes deployment manifests, ensuring all resource definitions convert correctly to JSON.

02 **Can I use `parse_yaml` if my file has multiple documents?**

Yes. The tool supports multi-document parsing, allowing you to process entire files containing several distinct configuration blocks separated by the standard '---' marker.

03 **What is the difference between this and a simple YAML converter?**

This MCP uses an industry-leading parser that passes complete official YAML test suites. It handles advanced features like aliases and anchors, which simpler converters often fail on.

04 **Is the output JSON structured correctly for my agent to use?**

The output is typed JSON designed for programmatic consumption. Because it validates against real specs, your agent receives data that's ready to be processed by subsequent steps in a workflow.

05 **Does this MCP support GitHub Actions workflows?**

Yes, you can use the `parse_yaml` tool to convert GitHub Actions workflow YAML into JSON. This is useful for programmatic validation of jobs and conditional expressions.

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 `"yaml-parser-engine": { "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

YAML Parser Engine 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 YAML Parser Engine. 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	YAML Parser Engine MCP
Server ID	`019e390f-b81f-7356-9c87-6362f983e1b6`
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/yaml-parser-engine.