MCP SERVER | NO CODE | CLOUD HOSTED

OpenRouteService MCP

Map routes and solve complex logistics problems.

OpenRouteService calculates complex routes and analyzes spatial data using OpenStreetMap data. It handles everything from basic directions for cars or bikes to solving multi-vehicle delivery problems, generating accurate reachability maps (isochrones), and finding the exact distance between multiple points.

A+ Quality Score 100/100

routing | geocoding | isochrones | distance-matrix | vehicle-routing-problem | spatial-analysis

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/openrouteservice

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.

OpenRouteService MCP

10 tools available
Cloud-hosted on Vinkius

Need to map out a logistical challenge? This MCP lets your AI client calculate optimal routes across any network type—car, bicycle, or walking path. You can generate detailed isochrone polygons showing exactly what areas are accessible within a set time limit, which is vital for urban planning. Beyond simple directions, you can compute distance matrices that map travel times between every combination of origins and destinations in your dataset. If you run delivery services, the VROOM solver handles complex vehicle routing problems with capacity constraints. Plus, if you only have raw GPS coordinates, the system cleans up noisy data points by snapping them to the nearest road segment. Connecting OpenRouteService through Vinkius gives your agent a single source for all these geospatial calculations.

Core Capabilities

01 — Plan multi-stop routes

Calculates optimal paths between multiple waypoints, providing distance and estimated time for different modes of travel.

02 — Map service coverage areas

Generates reachability polygons (isochrones) showing all points accessible within a specific driving or walking radius from a start point.

03 — Measure all-to-all distances

Creates matrices that compute the distance and duration between every origin and destination in a list of locations.

04 — Solve complex delivery logistics

Optimizes multi-vehicle routes, solving problems with limits on vehicle capacity or time windows.

05 — Standardize addresses and coordinates

Converts street addresses into precise coordinates, or vice versa, using reliable boundary filtering.

One Click on Vinkius — From Prompt to Execution

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

01 You ask your AI client to solve a spatial problem—for example, calculating routes for five different locations.

02 The MCP invokes the necessary calculation (like distance matrix or VRP optimization) using OpenStreetMap data in the background.

03 Your agent receives structured JSON output containing calculated distances, optimal paths, and detailed coordinates.

The bottom line is that your agent handles all the complex geographical calculations; you just ask it what problem needs solving.

Built For

Logistics managers who waste time manually plotting multi-stop routes, urban planners analyzing neighborhood accessibility, and field service coordinators needing to assign optimal technician paths.

Fleet Dispatch Manager

Uses the VROOM solver to determine the most efficient sequence of pickups and drop-offs for a day's worth of vehicles.

Urban Planner

Generates isochrone maps to visualize how far residents can walk or bike from transit centers within 15 minutes.

Site Operations Lead

Calculates the distance matrix between multiple warehouse zones and client sites to estimate total operational mileage.

What Changes When You Connect

01 Stop guessing travel time. Use `calculate_matrix` to get accurate duration and distance measurements between every pair of locations, making your estimates reliable.

02 Visualize service areas instead of just listing points. Generating isochrones with `calculate_isochrones` shows planners exactly what neighborhoods are within a 15-minute walk or drive.

03 Handle real-world logistics complexity. The VRP solver (`solve_vrp_optimization`) takes capacity and time limits into account, solving multi-stop delivery routes that simple mapping tools miss.

04 Clean up bad data instantly. If you collect messy GPS traces from a field team, `snap_gps_to_road` cleans the noise by snapping coordinates directly onto usable road segments.

05 Convert points to addresses and vice versa. Use `reverse_geocode` to take a random coordinate point and immediately get a human-readable street address for reporting.

Real-World Applications

Optimizing a multi-warehouse delivery network

A logistics manager needs the shortest total travel time for 12 deliveries using three different trucks. They ask their agent to run `solve_vrp_optimization`, which returns the optimal sequence and assignment, saving hours of manual spreadsheet work.

Assessing urban accessibility for a new development

An urban planner needs to know how many people can reach the site from major transit hubs within 10 minutes. They run `calculate_isochrones` centered on the hub, generating a precise polygon map instead of vague estimates.

Calculating total mileage for consulting sites

A field service lead has a list of 20 client addresses across three states. They use `calculate_matrix` to immediately get the distance and time between every single pair, allowing them to choose the most efficient cluster.

Cleaning up faulty GPS tracking data

A construction project generates noisy coordinates due to signal interference. The agent uses `snap_gps_to_road` on the raw feed, cleaning the points and making them usable for accurate distance calculations.

Patterns to Avoid

Trying to calculate routes manually

✗ AVOID

Copying 15 addresses into a standard mapping tool and hoping it plots the optimal path, which usually fails when dealing with capacity limits or time windows.

✓ INSTEAD

Don't use generic tools. Instead, ask your agent to run `calculate_matrix` for all origin/destination pairs first, then feed those results into the `solve_vrp_optimization` tool to get a truly optimized solution.

Assuming simple straight-line travel

✗ AVOID

Calculating the 'as the crow flies' distance between two points and assuming that is how long the drive will take.

✓ INSTEAD

Always use `calculate_directions` to get a route based on actual road networks. If you just need coordinates, run `geocode_search` first.

Misinterpreting data points

✗ AVOID

Receiving raw GPS coordinates and trying to calculate distances without understanding that the signal is noisy or inaccurate.

✓ INSTEAD

Run `snap_gps_to_road` on your collected data. This cleans the noise and ensures every point used for calculation sits accurately on a real road segment.

The Right Fit

Use this MCP if your problem involves measuring or planning movement across physical, measurable space. You need to know how long it takes to get from Point A to Point B via actual roads, not just the straight-line distance. If you are a planner dealing with logistics (deliveries, service zones), especially when managing multiple vehicles and time limits, this is your tool. Don't use it if you just need to look up an address in a database; simply run `reverse_geocode` or `geocode_search`. If you only need to know the geographical coordinates of a point without any routing logic, standard mapping APIs might suffice, but for anything involving travel time or network constraints, OpenRouteService is necessary.

Calculating distances and service areas used to be a manual nightmare.

Today, if you need to analyze coverage—say, where your field techs can get to within 20 minutes of the office—you spend hours opening multiple GIS platforms. You manually plot points, adjust boundaries, and calculate reachability zones one map at a time, often relying on outdated or simplified models.

With this MCP, you ask your agent for an isochrone map. It runs `calculate_isochrones` against real OpenStreetMap data, instantly generating the precise polygon showing every reachable area. You get actionable geospatial intelligence in seconds.

OpenRouteService gives you complete routing control.

Before this MCP, getting a comprehensive view of multiple destinations meant calculating distances separately for each pair and then manually plotting the connections. It was tedious copy-pasting between different mapping tools just to fill out an Excel sheet.

Now, you ask your agent to calculate the full distance matrix using `calculate_matrix`. You get one structured output that shows every single trip time and distance in a clean format. The whole planning process is done via conversation.

OpenRouteService: 10 Tools for Geospatial Data

These tools allow your agent to perform complex spatial calculations, including routing, boundary searches, and distance measurements, directly through the MCP.

#	TOOL	DESCRIPTION
01	`calculate_directions`	Finds precise, active routes spanning a road network between specified points.
02	`get_elevation_line`	Provides altitude data for a sequence of coordinates along a calculated line path.
03	`search_country_boundary`	Filters search results to ensure locations stay within specific national or regional borders.
04	`reverse_geocode`	Converts a set of coordinates back into a readable, structured address description.
05	`geocode_search`	Identifies specific locations or routes within the OpenRouteService system boundaries.
06	`calculate_isochrones`	Generates polygons that show all areas reachable from a central point within a defined time or distance.
07	`calculate_matrix`	Computes detailed tables showing the travel duration and distance between multiple pairs of locations (M x N).
08	`check_optimization_status`	Retrieves the current status or results of a previously initiated optimization job.
09	`solve_vrp_optimization`	Runs an automated solver to determine the optimal sequence and assignment for multiple vehicles on a set list of jobs.
10	`snap_gps_to_road`	Corrects noisy GPS data by forcing coordinates onto the nearest segment of a known road network.

See It in Action

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

U Calculate a driving route from Berlin to Munich with estimated time.

Route calculated (driving-car): Berlin → Munich. Distance: 585 km. Duration: 5h 42min. Route passes through Leipzig and Nuremberg. 2 toll sections detected. Want me to generate an isochrone from the destination or calculate the return route?

U Show me all areas reachable within 15 minutes by car from Times Square.

Isochrone generated for Times Square (40.7580, -73.9855) with 15-minute driving range. The polygon covers approximately 12 km² including Midtown, Upper East/West Side, Chelsea, and parts of the East Village. GeoJSON available. Want me to also generate 5 and 10-minute isochrones for comparison?

U Calculate the distance matrix between our 3 warehouses and 5 customer locations.

3×5 distance matrix computed (driving-car). Shortest pair: Warehouse B → Customer 3 (8.2 km, 14 min). Longest pair: Warehouse A → Customer 5 (47.1 km, 52 min). Average delivery time across all pairs: 28 min. Want me to run VRP optimization to assign optimal warehouse-customer pairs?

Frequently Asked Questions

01 **How do I use OpenRouteService to find the best route for multiple stops?**

You can calculate optimal routes using `solve_vrp_optimization`. This tool solves complex vehicle routing problems, assigning the most efficient sequence of jobs while respecting capacity and time limits.

02 **Is OpenRouteService better than general mapping APIs for distance calculations?**

Yes. While other services provide basic directions, this MCP offers `calculate_matrix` which computes detailed duration and distance between every single pair of points in a list, giving you comprehensive coverage.

03 **What if my GPS data is messy or noisy?**

Use the `snap_gps_to_road` tool. It cleans up coordinates by snapping them to the nearest actual road segment, making your raw field data usable for accurate routing.

04 **Can I find out what address corresponds to a set of latitude/longitude points?**

You use `reverse_geocode`. This tool takes coordinates and performs structural extraction, returning the corresponding readable street address details from OpenStreetMap boundaries.

05 **How do I check if an area is reachable within a certain time limit using OpenRouteService?**

Run `calculate_isochrones`. This generates a precise polygon showing every point accessible from your starting location within the specified time or distance, perfect for service zone analysis.

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 `"openrouteservice": { "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

OpenRouteService 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 OpenRouteService. 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	OpenRouteService MCP
Server ID	`019d75ec-dcb0-70be-81cf-3b5ccb298981`
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/openrouteservice.