AI BENCH LAB

MCP Server

Let your AI tools run benchmarks directly through the Model Context Protocol.

Version 1.0 | March 2026

aibenchlab.com

Reference · Professional AI Model Benchmarking for Windows

Contents

Let your AI tools run benchmarks directly through the Model Context Protocol.

1. Overview

The `aibenchlab-mcp` binary exposes AiBenchLab's benchmark capabilities as an MCP (Model Context Protocol) server. AI tools, agents, and coding assistants can discover models, trigger benchmarks, monitor progress, and export results — all through the standard MCP interface.

Requirements: Consultant tier or higher license.

Transport: Stdio (JSON-RPC 2.0)

2. Starting the Server

The MCP server communicates over stdin/stdout using the MCP stdio transport:

BASH

```
aibenchlab-mcp
```

Configure it in your AI tool's MCP settings. For example, in Claude Code's `settings.json`:

JSON

```
{
  "mcpServers": {
    "aibenchlab": {
      "command": "aibenchlab-mcp",
      "args": []
    }
  }
}
```

The server initializes the same backend as the desktop app (database, providers, model catalog, judge server) on startup.

3. Tools

The MCP server exposes 8 tools that AI agents can call.

list_models

List all AI models discovered across configured providers.

Parameters: None

Returns: Array of model objects with ID, name, provider, and metadata.

Example call:

JSON

```
{
  "name": "list_models",
  "arguments": {}
}
```

Example response:

JSON

```
[
  {"id": "ollama:llama3.2", "name": "llama3.2", "provider": "ollama"},
  {"id": "openai:gpt-4o", "name": "gpt-4o", "provider": "openai"}
]
```

---

list_providers

List all configured providers with status information.

Parameters: None

Returns: Array of provider configurations including name, type, endpoint, enabled status, and last connected timestamp.

---

list_sessions

List recent benchmark sessions.

Parameters:

Field	Type	Default	Description
`limit`	integer	20	Maximum sessions to return (max 50)

Returns: Array of recent sessions, most recent first. Each entry includes session ID, name, status, creation time, and summary scores.

---

run_benchmark

Add one or more models to the benchmark queue. Returns immediately — use `get_session` to monitor progress.

Parameters:

Field	Type	Required	Description
`model_ids`	string[]	Yes	Model IDs from `list_models`
`session_name`	string	No	Custom name (defaults to `MCP-{timestamp}`)
`tier`	string	No	`quick`, `standard`, or `comprehensive`
`domains`	string[]	No	Domain filter
`suite_id`	string	No	Specific test suite ID
`deterministic`	boolean	No	Enable deterministic execution
`seed`	integer	No	Global seed

Example call:

JSON

```
{
  "name": "run_benchmark",
  "arguments": {
    "model_ids": ["ollama:llama3.2"],
    "tier": "quick",
    "session_name": "agent-evaluation"
  }
}
```

Example response:

JSON

```
{
  "queued": 1,
  "items": [
    {
      "queue_item_id": "item-abc123",
      "run_config_id": "config-def456",
      "model_id": "ollama:llama3.2",
      "position": 1
    }
  ],
  "message": "Model(s) added to queue. Use get_session to monitor progress."
}
```

get_session

Fetch results for a benchmark session.

Parameters:

Field	Type	Required	Description
`session_id`	string	Yes	Session ID from `run_benchmark` or `list_sessions`

Returns:

- **While running**: Progress info — status, completion percentage, estimated time remaining.
- **When complete**: Full session results with composite Score, per-domain breakdowns, and individual test results.

Example call:

JSON

```
{
  "name": "get_session",
  "arguments": {
    "session_id": "abc123def456"
  }
}
```

---

get_queue

List items in the benchmark queue.

Parameters:

Field	Type	Default	Description
`include_done`	boolean	false	Include completed and failed items

Returns: Array of queue items with status (pending, running, completed, failed), model ID, position, and timing data.

---

cancel_queue_item

Cancel a pending queue item. Running items cannot be cancelled through this tool.

Parameters:

Field	Type	Required	Description
`queue_item_id`	string	Yes	Queue item ID from `get_queue`

Returns:

JSON

```
{
  "cancelled": true,
  "queue_item_id": "item-abc123"
}
```

If the item is not in a pending state:

JSON

```
{
  "cancelled": false,
  "reason": "Item not found or not in pending state (may already be running or completed)",
  "queue_item_id": "item-abc123"
}
```

---

export_report

Export a completed benchmark session to a file.

Parameters:

Field	Type	Required	Description
`session_id`	string	Yes	Session ID
`format`	string	Yes	`pdf`, `mbx`, `json`, or `csv`
`output_path`	string	Yes	Absolute file path. Parent directory must exist.

Returns:

JSON

```
{
  "success": true,
  "message": "Report exported to /path/to/report.pdf"
}
```

---

estimate_cost

Estimate benchmark cost before execution.

Parameters:

Field	Type	Required	Description
`model_name`	string	Yes	Model name
`provider_name`	string	Yes	Provider name
`provider_type`	string	Yes	`cloud` or `local`
`suite_id`	string	No	Suite ID (default: `standard`)

Returns: Full cost estimate with token counts, cost breakdown, confidence level, and operational projections. Same schema as the REST API's `/estimate-cost` endpoint.

---

4. Resources

The MCP server also exposes readable resources:

URI	Description
`aibl://sessions`	List of recent benchmark sessions
`aibl://sessions/{id}`	Full results for a specific session
`aibl://models`	All discovered AI models
`aibl://providers`	Provider configuration and status

Resources are read-only and accessed via the standard MCP `read_resource` method.

---

5. Valid Domain Names

Use these with the `domains` parameter in `run_benchmark`:

```
reasoning        code             chat
tool_calling     adversarial_safety   deployment_risk
agentic          multimodal       multi_turn_adversarial
agentic_email    context_retention
```

---

6. Example Workflows

AI Agent: Evaluate a model before deployment

An AI agent deciding which model to deploy can:

› **List available models** with `list_models`
› **Estimate cost** with `estimate_cost` for each candidate
› **Run benchmarks** with `run_benchmark` against a production-readiness suite
› **Poll for results** with `get_session`
› **Export the report** with `export_report` for the team to review

Automated regression testing

A CI/CD pipeline can:

› **Run a benchmark** with `run_benchmark` using `deterministic: true` and a fixed seed
› **Poll until complete** with `get_session`
› **Compare the Score** against a baseline threshold
› **Export results** with `export_report` for archival

Cost-aware model selection

An agent managing API costs can:

› **Estimate costs** for multiple models with `estimate_cost`
› **Select the cheapest model** that meets a quality threshold
› **Run a quick benchmark** to verify quality
› **Report findings** with score and cost data

---

7. Configuration

The MCP server uses the same configuration as the desktop app:

- **Database**: `~/.aibenchlab/benchmarks.db`
- **Providers**: Auto-detected (Ollama, LM Studio) and user-configured (cloud APIs)
- **Components**: Must be installed via the desktop app's Setup screen

Environment variables:

Variable	Default	Description
`RUST_LOG`	`info`	Log verbosity (logs go to stderr)

8. Prerequisites

Before using the MCP server:

› **Install AiBenchLab** (desktop app) and complete the Setup wizard to download required components (llama.cpp runtime, judge model).

› **Configure providers** — add API keys for cloud providers through the desktop app's Settings.

› **Verify models** — ensure at least one model is available (`list_models` should return results).

The MCP server does not include a component installer or provider configuration UI. All setup is done through the desktop app.

Resources

• Documentation: **aibenchlab.com/docs**
• Support: **aibenchlab.com/contact**
• YouTube walkthroughs: **youtube.com/@aibenchlab**
• Email: **support@aibenchlab.com**

AiBenchLab is built by a solo founder with 40+ years of software engineering experience. Every feature is designed with the conviction that you deserve honest, transparent, and reproducible AI model evaluations. No hype. Just truth.