AI BENCH LAB

CLI Reference

Run benchmarks from the command line. Automate evaluation in scripts and CI/CD pipelines.

Version 1.0 | March 2026

aibenchlab.com

Reference · Professional AI Model Benchmarking for Windows

Contents

Run benchmarks from the command line. Automate evaluation in scripts and CI/CD pipelines.

1. Overview

The `aibenchlab-cli` binary provides headless benchmark execution, result export, model listing, and cost estimation. It shares the same backend as the desktop app but runs without a GUI.

Requirements: Pro tier or higher license.

2. Commands

`run`

Run a benchmark session.

BASH

```
aibenchlab-cli run [FLAGS] [OPTIONS]
```

Options:

Flag	Type	Description
`--model <ID>`	String (repeatable)	Model ID in `provider:model` format. Specify multiple times for multi-model runs.
`--model-file <PATH>`	Path	Load model IDs from a file (one per line).
`--domain <NAME>`	String (repeatable)	Restrict to specific domains. Specify multiple times to include several.
`--language <LANG>`	String (repeatable)	Language filter.
`--tier <TIER>`	String	Test tier: `quick`, `standard`, or `comprehensive`.
`--suite_id <ID>`	String	Run a specific test suite by ID.
`--suite_run_id <ID>`	String	Associate with a suite run ID.
`--session_name <NAME>`	String	Custom session name (defaults to timestamp).

Flag	Type	Description
`--deterministic`	Flag	Enable deterministic execution (fixed order, no sampling randomness).
`--seed <N>`	u64	Global seed for deterministic mode.
`--temperature <T>`	f32	Temperature override for all provider requests.
`--export_format <FMT>`	String	Auto-export after run: `pdf`, `mbx`, `json`, or `csv`.
`--export_path <PATH>`	Path	Export file path (defaults to `./aibenchlab-{session_id}.{ext}`).
`--output_json`	Flag	Output a machine-readable JSON summary to stdout.
`--cost_report <PATH>`	Path	Export a cost analysis report (JSON) after the run completes.

Examples:

Run a benchmark against a local Ollama model:

BASH

```
aibenchlab-cli run --model ollama:llama3.2
```

Run against multiple models with domain filtering:

BASH

```
aibenchlab-cli run \
  --model ollama:llama3.2 \
  --model openai:gpt-4o \
  --domain reasoning \
  --domain coding
```

Run a specific suite with JSON export:

BASH

```
aibenchlab-cli run \
  --model ollama:mistral \
  --suite_id quick-compare \
  --export_format json \
  --export_path ./results.json
```

Deterministic run with a fixed seed:

BASH

```
aibenchlab-cli run \
  --model ollama:llama3.2 \
  --deterministic \
  --seed 42
```

Machine-readable output for scripting:

BASH

```
aibenchlab-cli run --model ollama:llama3.2 --output_json
```

Output (stdout):

JSON

```
{
  "session_id": "abc123def456",
  "status": "completed"
}
```

Run with a model list file:

BASH

```
# models.txt
ollama:llama3.2
ollama:mistral
openai:gpt-4o-mini
```

BASH

```
aibenchlab-cli run --model-file models.txt --tier comprehensive
```

Run with cost report:

BASH

```
aibenchlab-cli run \
  --model openai:gpt-4o \
  --cost_report ./cost-analysis.json
```

`export`

Export a completed benchmark session to a file.

BASH

```
aibenchlab-cli export --session_id &lt;ID&gt; --format &lt;FMT&gt; --output &lt;PATH&gt;
```

Options:

Flag	Type	Required	Description
`--session_id <ID>`	String	Yes	Session ID to export.
`--format <FMT>`	String	Yes	Export format: `pdf`, `mbx`, `json`, or `csv`.
`--output <PATH>`	Path	Yes	Output file path.

Example:

BASH

```
aibenchlab-cli export \
  --session_id abc123def456 \
  --format pdf \
  --output ./benchmark-report.pdf
```

`list-models`

List all available AI models across configured providers.

BASH

```
aibenchlab-cli list-models
```

Output (one model per line):

```
ollama:llama3.2 (ollama)
ollama:mistral (ollama)
openai:gpt-4o (openai)
openai:gpt-4o-mini (openai)
anthropic:claude-3-5-sonnet-20241022 (anthropic)
```

`estimate-cost`

Estimate the cost of a benchmark run without executing it.

BASH

```
aibenchlab-cli estimate-cost --model &lt;ID&gt; [OPTIONS]
```

Options:

Flag	Type	Required	Description
`--model <ID>`	String	Yes	Model ID in `provider:model` format.
`--provider_type <TYPE>`	String	No	`cloud` or `local` (auto-detected if omitted).
`--suite_id <ID>`	String	No	Suite to estimate for (default: `standard`).

Example:

BASH

```
aibenchlab-cli estimate-cost --model openai:gpt-4o
```

Output is a JSON cost estimate printed to stderr.

---

3. Valid Domain Names

Use these with the `--domain` flag:

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

---

4. Valid Tier Names

Use these with the `--tier` flag:

Tier	Description
`quick`	Subset of tests for fast iteration
`standard`	Balanced coverage (default)
`comprehensive`	Full test suite — all 254 tests

---

5. Export Formats

Format	Description
`pdf`	Professional PDF report
`mbx`	AiBenchLab package format (importable)
`json`	Machine-readable JSON results
`csv`	Spreadsheet-compatible CSV

6. Output Behavior

- **User-facing output** (model lists, JSON summaries) goes to **stdout**.
- **Progress, logs, and errors** go to **stderr** via the standard Rust logger.
- Set the `RUST_LOG` environment variable to control log verbosity:

BASH

```
RUST_LOG=info aibenchlab-cli run --model ollama:llama3.2
RUST_LOG=debug aibenchlab-cli run --model ollama:llama3.2
```

7. Initialization

On startup, the CLI performs the same initialization as the desktop app:

› Storage directory creation
› SQLite database init (WAL mode, foreign keys enabled)
› Schema migration
› Provider registry and auto-detection (Ollama, LM Studio, LocalAI)
› Model catalog loading
› Hardware scan (GPU detection)
› Judge server startup

The judge model and llama.cpp runtime must be installed via the desktop app's Setup screen before the CLI can run benchmarks. The CLI does not include a component installer.

8. CI/CD Integration

GitHub Actions Example

YAML

```yaml
name: AI Model Benchmark
on:
  schedule:
    - cron: '0 6 * * 1'  # Weekly Monday 6am
  workflow_dispatch:

jobs:
  benchmark:
    runs-on: self-hosted  # Requires GPU runner with AiBenchLab installed
    steps:
      - name: Run benchmark
        run: |
          aibenchlab-cli run \
            --model ollama:llama3.2 \
            --suite_id quick-compare \
            --deterministic \
            --seed 42 \
            --export_format json \
            --export_path ./results.json \
            --output_json &gt; summary.json

      - name: Upload results
        uses: actions/upload-artifact@v4
        with:
          name: benchmark-results
          path: |
            results.json
            summary.json
```

Scripted Comparison

BASH

```bash
#!/bin/bash
# Compare two models and export results

MODELS=("ollama:llama3.2" "ollama:mistral")

for model in "${MODELS[@]}"; do
  echo "Benchmarking $model..."
  aibenchlab-cli run \
    --model "$model" \
    --suite_id quick-compare \
    --deterministic --seed 42 \
    --export_format json \
    --export_path "./results-$(echo $model | tr ':' '-').json" \
    --output_json
done

echo "Done. Results saved to ./results-*.json"
```

9. Exit Codes

Code	Meaning
0	Success
1	Runtime error (benchmark failure, export error, etc.)
2	Invalid arguments

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.