Use LlamaIndex with Radium

LlamaIndex provides `OpenAILike` for third-party OpenAI-compatible APIs. Use it to connect a LlamaIndex application to Radium.

Before you start

You need Python 3.10 or newer, a Radium API key, and a terminal or PowerShell window.

1. Create a project and install LlamaIndex

macOS or Linux:

```
mkdir radium-llamaindex
cd radium-llamaindex
python3 -m venv .venv
source .venv/bin/activate
python -m pip install llama-index-llms-openai-like
```

Windows PowerShell:

```
mkdir radium-llamaindex
cd radium-llamaindex
py -m venv .venv
.venv\Scripts\Activate.ps1
python -m pip install llama-index-llms-openai-like
```

2. Set your Radium credentials

macOS or Linux:

```
export RADIUM_API_KEY="your-radium-api-key"
export RADIUM_MODEL="hal-1.0"
```

Windows PowerShell:

```
$env:RADIUM_API_KEY = "your-radium-api-key"
$env:RADIUM_MODEL = "hal-1.0"
```

3. Create `main.py`

```
import os

from llama_index.core.llms import ChatMessage
from llama_index.llms.openai_like import OpenAILike



api_key = os.getenv("RADIUM_API_KEY")
if not api_key:
    raise SystemExit("Set RADIUM_API_KEY before running this program.")

llm = OpenAILike(
    model=os.getenv("RADIUM_MODEL", "hal-1.0"),
    api_key=api_key,
    api_base="https://api.radium.cloud/v1",
    is_chat_model=True,
    max_tokens=512,
    temperature=0,
    timeout=90,
)

response = llm.chat(
    [
        ChatMessage(role="system", content="Follow the user's instruction exactly."),
        ChatMessage(role="user", content="Reply with exactly: Radium connected."),
    ]
)

content = response.message.content
if not content or not content.strip():
    raise RuntimeError("Radium returned no visible text.")
print(f"RADIUM_RESPONSE: {content.strip()}")
```

LlamaIndex calls the endpoint setting `api_base`, not `base_url`.

4. Run it

```
python main.py
```

Expected output:

```
RADIUM_RESPONSE: Radium connected.
```

Choose a model

```
export RADIUM_MODEL="clarke-1.0"  # or hal-1.0 or tycho-1.0
python main.py
```

Migrate an existing LlamaIndex app

Replace its current LLM object:

```
llm = OpenAILike(
    model="hal-1.0",
    api_key=os.environ["RADIUM_API_KEY"],
    api_base="https://api.radium.cloud/v1",
    is_chat_model=True,
    max_tokens=512,
)
```

Pass `llm` to the existing index, query engine, workflow, or agent. A RAG application still needs a separate embedding model; do not use these chat model IDs as embedding model IDs.

Troubleshooting

- Authentication errors: check `RADIUM_API_KEY` in the active terminal.
- `404`: keep `/v1` in `api_base` and use an exact Radium model ID.
- Empty output: use at least 512 output tokens.
- Import errors: activate `.venv` and reinstall the pinned package.

Validation

Version note: these instructions were verified with Python 3.12.7 and `llama-index-llms-openai-like==0.7.2` on August 20, 2026. You do not need that exact package version. Text generation and `FunctionAgent` tool calling passed with `hal-1.0`, `clarke-1.0`, and `tycho-1.0`.

Reference: LlamaIndex OpenAILike integration.

Next

- API quickstart, for calling Radium directly
- Tool calling, for the `tool_use` and `tool_result` contract

Created 2026-08-23 21:36:36 UTC by Admin
Updated 2026-08-23 21:53:48 UTC by Admin