Use LangChain with Radium

LangChain's `ChatOpenAI` class works with OpenAI-compatible APIs. Point it at Radium and continue using normal LangChain messages, chains, and agents.

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 LangChain

macOS or Linux:

```
mkdir radium-langchain
cd radium-langchain
python3 -m venv .venv
source .venv/bin/activate
python -m pip install langchain-openai
```

Windows PowerShell:

```
mkdir radium-langchain
cd radium-langchain
py -m venv .venv
.venv\Scripts\Activate.ps1
python -m pip install langchain-openai
```

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 langchain_openai import ChatOpenAI



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

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

response = llm.invoke(
    [
        ("system", "Follow the user's instruction exactly."),
        ("human", "Reply with exactly: Radium connected."),
    ]
)

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

4. Run it

```
python main.py
```

Expected output:

```
RADIUM_RESPONSE: Radium connected.
```

Choose a model

Use any validated Radium model:

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

Migrate an existing LangChain app

Replace the model initialization used by your chain or agent:

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

Your prompts, chains, tools, and message objects can remain unchanged. This integration uses Chat Completions, so do not enable `use_responses_api`.

Troubleshooting

- Authentication errors: set `RADIUM_API_KEY` in the terminal running the program.
- `404`: keep `/v1` in `base_url` and use an exact Radium model ID.
- Empty output: use `max_tokens=512` or higher.
- Import errors: reactivate `.venv` and reinstall the pinned package.

Validation

Version note: these instructions were verified with Python 3.12.7 and `langchain-openai==1.6.0` on August 20, 2026. You do not need that exact package version. Text generation and bound-tool

calling passed with `hal-1.0`, `clarke-1.0`, and `tycho-1.0`.

Reference: LangChain ChatOpenAI documentation.

Next

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

Created 2026-08-23 21:35:44 UTC by Admin
Updated 2026-08-23 21:53:29 UTC by Admin