Use CrewAI with Radium

CrewAI's `LLM` class uses LiteLLM for model connections. Select the OpenAI-compatible protocol with an `openai/` prefix and send requests 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 CrewAI

macOS or Linux:

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

Windows PowerShell:

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

2. Set your Radium credentials

macOS or Linux:

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

Windows PowerShell:

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

Tracing is disabled here so this connection example does not ask for an additional tracing setup.

3. Create `main.py`

```
import os

from crewai import Agent, Crew, LLM, Task



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

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

assistant = Agent(
    role="Radium connection tester",
    goal="Follow the user's instruction exactly",
    backstory="You provide short, exact responses.",
    llm=llm,
    allow_delegation=False,
    verbose=False,
)
task = Task(
    description="Reply with exactly: Radium connected.",
    expected_output="The exact text 'Radium connected.' and nothing else.",
    agent=assistant,
```

```
)

result = Crew(agents=[assistant], tasks=[task], verbose=False).kickoff()
if not result.raw or not result.raw.strip():
    raise RuntimeError("Radium returned no visible text.")
print(f"RADIUM_RESPONSE: {result.raw.strip()}")
```

The `openai/` prefix is required for LiteLLM provider routing. Radium still receives the model ID without that prefix.

4. Run it

```
python main.py
```

Expected final output:

```
RADIUM_RESPONSE: Radium connected.
```

CrewAI may print status messages before the final line.

Choose a model

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

Migrate an existing crew

Create one Radium-backed LLM and pass it to every agent that should use Radium:

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

agent = Agent(..., llm=llm)
```

Your roles, goals, tasks, and crew process can remain unchanged. Planning, memory, and external tools may require additional model calls or credentials.

Troubleshooting

- Authentication errors: set the API key in the same terminal running CrewAI.
- Provider errors: keep the `openai/` prefix on the model passed to `LLM`.
- `404`: use the exact base URL and model IDs shown above.
- Unexpected tracing prompts: set `CREWAI_TRACING_ENABLED=false`.

Validation

Version note: these instructions were verified with Python 3.12.7 and `crewai==1.15.17` on August 20, 2026. You do not need that exact package version. Agent/task generation and CrewAI tool execution passed with `hal-1.0`, `clarke-1.0`, and `tycho-1.0`.

Reference: CrewAI custom LLM guide.

Next

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

Created 2026-08-23 21:34:54 UTC by Admin
Updated 2026-08-23 21:53:09 UTC by Admin