Use PydanticAI with Radium

PydanticAI supports custom OpenAI-compatible providers. Construct an `OpenAIChatModel` explicitly so requests use Radium's Chat Completions endpoint.

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 PydanticAI

macOS or Linux:

```
mkdir radium-pydanticai
cd radium-pydanticai
python3 -m venv .venv
source .venv/bin/activate
python -m pip install "pydantic-ai-slim[openai]"
```

Windows PowerShell:

```
mkdir radium-pydanticai
cd radium-pydanticai
py -m venv .venv
.venv\Scripts\Activate.ps1
python -m pip install "pydantic-ai-slim[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 pydantic_ai import Agent
from pydantic_ai.models.openai import OpenAIChatModel
from pydantic_ai.providers.openai import OpenAIProvider



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

model = OpenAIChatModel(
    os.getenv("RADIUM_MODEL", "hal-1.0"),
    provider=OpenAIProvider(
        api_key=api_key,
        base_url="https://api.radium.cloud/v1",
    ),
)
agent = Agent(
    model,
    system_prompt="Follow the user's instruction exactly.",
    model_settings={"max_tokens": 512, "temperature": 0},
)

result = agent.run_sync("Reply with exactly: Radium connected.")
if not result.output or not str(result.output).strip():
    raise RuntimeError("Radium returned no visible text.")
print(f"RADIUM_RESPONSE: {str(result.output).strip()}")
```

Do not use the shorthand `openai:hal-1.0`. Constructing `OpenAIChatModel` explicitly ensures PydanticAI uses the custom Radium provider and Chat Completions endpoint.

4. Run it

```
python main.py
```

Expected output:

```
RADIUM_RESPONSE: Radium connected.
```

Choose a model

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

Migrate an existing PydanticAI agent

Replace its model/provider configuration:

```
model = OpenAIChatModel(
    "hal-1.0",
    provider=OpenAIProvider(
        api_key=os.environ["RADIUM_API_KEY"],
        base_url="https://api.radium.cloud/v1",
    ),
)
agent = Agent(model)
```

Your system prompts, dependency types, tools, and output types can remain unchanged.

Troubleshooting

- Authentication errors: set the key in the terminal running Python.
- Calls going to OpenAI instead of Radium: use the explicit `OpenAIProvider` construction shown above.
- `404`: verify `/v1` and the exact model name.
- Empty output: keep the output allowance at 512 or higher.

Validation

Version note: these instructions were verified with Python 3.12.7 and `pydantic-ai-slim[openai]==2.31.1` on August 20, 2026. You do not need that exact package version. Text

generation and `tool_plain` execution passed with `hal-1.0`, `clarke-1.0`, and `tycho-1.0`.

Reference: PydanticAI OpenAI-compatible models.

Next

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

Created 2026-08-23 21:37:36 UTC by Admin
Updated 2026-08-23 21:54:14 UTC by Admin