



Providers

Litellm

LiteLLM is an open-source LLM gateway that provides a unified API to 100+ model providers. Route OpenClaw through LiteLLM to get centralized cost tracking, logging, and the flexibility to switch backends without changing your OpenClaw config.

Why use LiteLLM with OpenClaw?

Cost tracking – See exactly what OpenClaw spends across all models

Model routing – Switch between Claude, GPT-4, Gemini, Bedrock without config changes

Virtual keys – Create keys with spend limits for OpenClaw

Logging – Full request/response logs for debugging

Fallbacks – Automatic failover if your primary provider is down

Quick start

Via onboarding

```
openclaw onboard --auth-choice litellm-api-key
```

Manual setup

1. Start LiteLLM Proxy:



```
pip install 'litellm[proxy]'  
litellm --model claude-opus-4-6
```

2. Point OpenClaw to LiteLLM:

```
export LITELLM_API_KEY="your-litellm-key"  
  
openclaw
```

That's it. OpenClaw now routes through LiteLLM.

Configuration

Environment variables

```
export LITELLM_API_KEY="sk-litellm-key"
```

Config file



```
models: {
  providers: {
    litellm: {
      baseUrl: "http://localhost:4000",
      apiKey: "${LITELLM_API_KEY}",
      api: "openai-completions",
      models: [
        {
          id: "claude-opus-4-6",
          name: "Claude Opus 4.6",
          reasoning: true,
          input: ["text", "image"],
          contextWindow: 200000,
          maxTokens: 64000,
        },
        {
          id: "gpt-4o",
          name: "GPT-4o",
          reasoning: false,
          input: ["text", "image"],
          contextWindow: 128000,
          maxTokens: 8192,
        },
      ],
    },
  },
  agents: {
    defaults: {
      model: { primary: "litellm/claude-opus-4-6" },
    },
  },
}
```

Virtual keys

Create a dedicated key for OpenClaw with spend limits:

```
 curl -X POST "http://localhost:4000/key/generate" \  
-H "Authorization: Bearer $LITELLM_MASTER_KEY" \  
-H "Content-Type: application/json" \  
-d '{  
  "key_alias": "openclaw",  
  "max_budget": 50.00,  
  "budget_duration": "monthly"  
}
```

Use the generated key as `LITELLM_API_KEY` .

Model routing

LiteLLM can route model requests to different backends. Configure in your LiteLLM `config.yaml` :

```
model_list:  
  - model_name: claude-opus-4-6  
    litellm_params:  
      model: claude-opus-4-6  
      api_key: os.environ/ANTHROPIC_API_KEY  
  
  - model_name: gpt-4o  
    litellm_params:  
      model: gpt-4o  
      api_key: os.environ/OPENAI_API_KEY
```

OpenClaw keeps requesting `claude-opus-4-6` – LiteLLM handles the routing.

Viewing usage

Check LiteLLM's dashboard or API:

```
# Key info
curl "http://localhost:4000/key/info" \
  -H "Authorization: Bearer sk-litellm-key"

# Spend logs
curl "http://localhost:4000/spend/logs" \
  -H "Authorization: Bearer $LITELLM_MASTER_KEY"
```

Notes

LiteLLM runs on `http://localhost:4000` by default

OpenClaw connects via the OpenAI-compatible `/v1/chat/completions` endpoint

All OpenClaw features work through LiteLLM – no limitations

See also

[OpenRouter](#)

[Amazon Bedrock](#)

Powered by [mintlify](#)