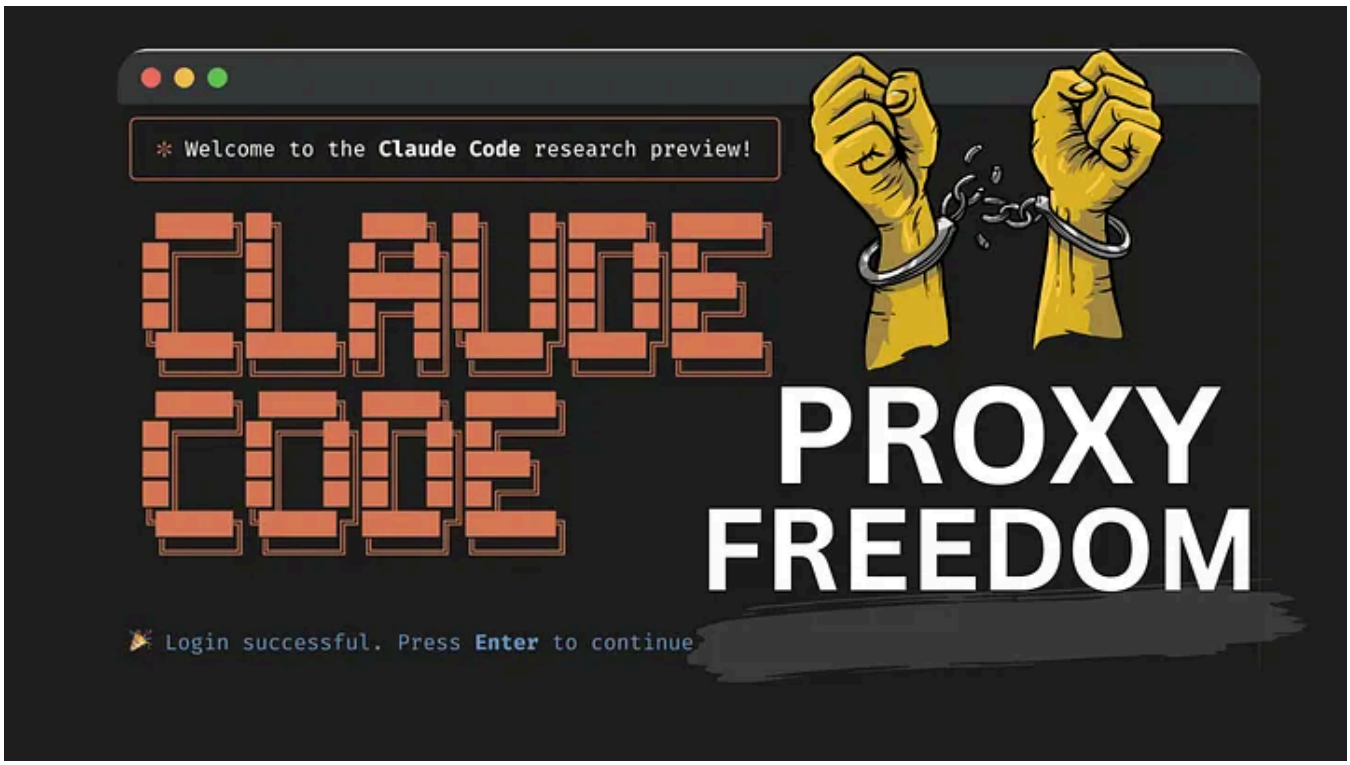
How I Use Claude Code & Proxies To Unlock My AI Coding Freedom



Claude Code & Proxies Tutorial — Featured Image

What if I told you there's a way to make Claude Code work in ANY environment, bypass the restriction, and even use completely different AI models while keeping the same familiar interface...

But most developers have no clue that these proxy techniques exist.

> ***There's a way to make Claude Code work with completely different AI models while keeping the same familiar interface.***

Proxies with Claude Code aren't just about network routing.

They're your gateway to using DeepSeek, Chinese models, GPT-4o, Gemini, and hundreds of other models through Claude Code's interface.

But you need to understand what you're doing, or you might break your entire setup.

Think about it.

You love Claude Code's workflow, but hit usage limits at the worst times.

> ***Or you want to use DeepSeek's reasoning capabilities for complex debugging while keeping Claude Code's excellent tool integration.***

Maybe you need access to specialized Chinese models like Kimi K2 or want to run everything through free models to cut costs dramatically.

That's where proxy magic comes in.

I've been testing every Claude Code proxy method I can find.

The simple `ANTHROPIC_BASE_URL` redirect that lets you use any OpenAI-compatible API. Advanced routing systems like Claude Code Router automatically pick the best model for each task.

Cost-saving setups using DeepSeek that cut expenses by 96%.

Some techniques unlock incredible capabilities. Others are pitfalls that will leave you debugging proxy chains instead of coding.

> I have spent countless hours testing and dealing with proxy headaches, and I've learned a lot about what works. I have even built my own proxy apps that extend Claude Code. You can learn a lot from my other Claude Code tutorials here.

The proxy ecosystem for Claude Code is vast.

You can route different coding tasks to different AI models, maintain Claude Code's interface while using completely different providers, set up intelligent cost optimization, and access models that Anthropic doesn't even offer.

Ready to unlock the hidden potential of Claude Code proxies?

> ***I'll walk through the methods that actually work, starting with the simplest redirects and building up to advanced multi-model routing systems.***

Simple Model Switching Magic

The easiest way to use different models with Claude Code is the `ANTHROPIC_BASE_URL` environment variable.

This single setting redirects all Claude Code requests to any OpenAI-compatible API.

```
export ANTHROPIC_BASE_URL=https://api.deepseek.com/anthropic
export ANTHROPIC_AUTH_TOKEN=your-deepseek-key
claude
```

Claude Code thinks it's talking to Anthropic, but you're using DeepSeek's reasoning models at a fraction of the cost.

DeepSeek Integration

DeepSeek offers direct Anthropic API compatibility. Their reasoning model handles complex debugging better than most alternatives, and the cost difference is massive.

```
export ANTHROPIC_BASE_URL=https://api.deepseek.com/anthropic
export ANTHROPIC_MODEL=deepseek-chat
export API_TIMEOUT_MS=600000  # DeepSeek can take time for complex reasoning
```

OpenRouter for Everything Else

Want access to 400+ models through one API? OpenRouter is your Swiss Army knife.

```
export ANTHROPIC_BASE_URL=https://openrouter.ai/api/v1
export ANTHROPIC_AUTH_TOKEN=your-openrouter-key
```

Now you can use GPT-4o, Gemini, Qwen3-Coder, or even free models `qwen/qwen3-coder:free` that cost literally nothing for routine tasks.

Chinese Models Access

Getting access to Chinese models like Kimi K2 is straightforward through OpenRouter:

```
# Access Kimi K2 through OpenRouter
export ANTHROPIC_BASE_URL=https://openrouter.ai/api/v1
export ANTHROPIC_AUTH_TOKEN=your-key
# Then use model: moonshotai/kimi-k2
```

Or directly through Moonshot AI:

```
export ANTHROPIC_BASE_URL=https://api.moonshot.ai/anthropic
export ANTHROPIC_AUTH_TOKEN=your-moonshot-key
```

The beauty is that Claude Code's interface remains identical.

Same commands, same workflow, completely different AI models powering your coding sessions.

Advanced Proxy Systems

Simple redirects are just the beginning.

> ***Advanced proxy systems let you route different types of requests to different models automatically, optimize costs, and create failover systems.***

Claude Code Router — The Smart Proxy

Claude Code Router intercepts your requests and routes them based on task complexity.

```
npm install -g @musistudio/claude-code-router
```

Create `~/.claude-code-router/config.json`:

```
{
  "Providers": [
    {
      "name": "deepseek",
      "api_base_url": "https://api.deepseek.com/chat/completions",
      "api_key": "your-key",
      "models": ["deepseek-chat", "deepseek-reasoner"]
    },
    {
      "name": "openrouter",
      "api_base_url": "https://openrouter.ai/api/v1/chat/completions",
      "api_key": "your-key",
      "models": ["qwen/qwen3-coder:free", "anthropic/claude-3.5-sonnet"]
    }
  ],
  "Router": {
    "default": "openrouter,qwen/qwen3-coder:free",
    "think": "deepseek,deepseek-reasoner",
    "background": "openrouter,qwen/qwen3-coder:free"
  }
}
```

Now, simple tasks use free models, complex reasoning gets routed to DeepSeek, and you can switch models mid-conversation with `/model deepseek,deepseek-reasoner`.

LiteLLM Integration

For enterprise setups, LiteLLM provides centralized model management with usage tracking and cost controls.

```yaml
model_list:
  - model_name: deepseek-chat
    litellm_params:
      model: deepseek/deepseek-chat
      api_key: os.environ/DEEPSEEK_API_KEY
  - model_name: kimi-k2
    litellm_params:
      model: moonshot/kimi-k2
      api_key: os.environ/MOONSHOT_API_KEY
```

Then connect Claude Code:

```bash
export ANTHROPIC_BASE_URL=http://localhost:4000
export ANTHROPIC_AUTH_TOKEN=your-litellm-key
```

Cost Optimization Routing

Set up intelligent cost routing that automatically picks cheaper models for simple tasks:

```json
{
  "Router": {
    "default": "openrouter,qwen/qwen3-coder:free",
    "complex_reasoning": "deepseek,deepseek-reasoner",
    "speed_critical": "groq,mixtral-8x7b-32768",
    "budget_conscious": "openrouter,google/gemini-2.0-flash"
  },
  "cost_limits": {
    "daily_budget": 10.0,
    "per_session_limit": 2.0
  }
}
```

This setup can reduce your AI coding costs by up to 95% while maintaining quality for tasks that need it.

Specialized Proxy Setups

Beyond basic routing, there are specialized proxy configurations for specific use cases — local models, hybrid setups, and debugging configurations that solve real problems.

Local Models with Ollama

Running models locally gives you complete privacy and zero API costs.

Ollama provides OpenAI-compatible endpoints that work directly with Claude Code.

```
# Start Ollama with a coding model
ollama serve
ollama pull qwen2.5-coder:32b
```

```
# Connect Claude Code
export ANTHROPIC_BASE_URL=http://localhost:11434/v1
export ANTHROPIC_AUTH_TOKEN=ollama
```

The trade-off is speed and capability, but for sensitive codebases or when you're offline, local models are invaluable.

Hybrid Cloud/Local Setup

The smartest approach combines local models for routine tasks with cloud models for complex reasoning:

```
{
  "Providers": [
    {
      "name": "ollama",
      "api_base_url": "http://localhost:11434/v1/chat/completions",
      "models": ["qwen2.5-coder:32b"]
    },
    {
      "name": "deepseek",
      "api_base_url": "https://api.deepseek.com/chat/completions",
```

```
      "models": ["deepseek-reasoner"]
    }
  ],
  "Router": {
    "default": "ollama,qwen2.5-coder:32b",
    "complex_reasoning": "deepseek,deepseek-reasoner"
  }
}
```

Debugging Proxy Issues

I discovered this trick when proxies break, and debugging becomes critical.

Enable detailed logging:

```
export ANTHROPIC_LOG=debug
export API_TIMEOUT_MS=600000
```

```
# Test connectivity
curl -I --proxy $HTTPS_PROXY https://api.deepseek.com
```

Common issues and fixes:

- **SSL Certificate problems**: Set `NODE_TLS_REJECT_UNAUTHORIZED=0` (only in trusted networks)
- **Timeout issues**: Increase `API_TIMEOUT_MS` for slower models
- **Authentication failures**: Check API key formats for different providers

Multi-Provider Failover

Set up automatic failover when primary providers hit rate limits:

```
{
  "Providers": [
    {"name": "primary", "models": ["deepseek-chat"]},
    {"name": "backup", "models": ["qwen/qwen3-coder:free"]},
    {"name": "emergency", "models": ["ollama,qwen2.5-coder:7b"]}
```

```
    ],
    "failover_strategy": "cascade"
}
```

This ensures you never lose coding flow due to API limits or service outages.

Real-World Integration

After months of testing different proxy configurations, here's what works in daily development and what you should avoid.

My Daily Setup

I run a three-tier system that handles 95% of my coding needs :

```
# Tier 1: Free models for routine tasks
export ANTHROPIC_BASE_URL=https://openrouter.ai/api/v1
export ANTHROPIC_MODEL=qwen/qwen3-coder:free
```

```
# Tier 2: DeepSeek for complex reasoning
# Tier 3: Claude Opus for architecture decisions
```

The cost breakdown is dramatic.

> ***What used to cost $50/month now runs under $5, with better performance for specific tasks.***

What Works Long-Term

Router configurations that automatically handle model selection save the most time. Manual switching gets tedious fast.

Dynamic model routing based on context works.

Simple text analysis routes to free models, function calling gets proper tool-capable models, and complex debugging goes to reasoning specialists.

DeepSeek integration has been rock solid. Their Anthropic API compatibility means zero configuration headaches, and the reasoning quality rivals much more expensive alternatives.

What Doesn't Work

> ***Overly complex routing rules create debugging nightmares. Keep routing logic simple and predictable.***

Disabling SSL verification is tempting for quick fixes, but it creates security holes. Always use proper certificate handling.

Multiple proxy layers rarely add value and often create timeout issues.

Final Thoughts

Claude Code proxies aren't just about cost savings. In my view, it's about having options when you need them.

When Anthropic has outages, your workflow continues. When you hit usage limits, free alternatives keep you coding.

When you need specialized reasoning or multilingual support, Chinese models deliver capabilities ***that Claude doesn't offer — though I always give a Caveat when using a Chinese model check without first checking the organization's privacy guidelines.***

In my experience using proxies, the setup complexity is front-loaded. Once your routing works, it becomes invisible infrastructure that handles model selection smoothly.

> ***If you are new to using proxies with Claue Code, start simple with direct*** `ANTHROPIC_BASE_URL` ***redirects to understand the basics. Add routing complexity only when you need it. Focus on models that solve problems in your workflow rather than collecting options.***

The proxy ecosystem around Claude Code transforms it from a single-provider tool into a universal AI coding interface. That flexibility is worth giving it a try!

> *You can check out specific tutorials I have covered before for setting up Claude Cide with specific models using proxies.*

Here is the collection of all my Claude Code tutorials.

If you enjoyed this article, ***be sure to follow my upcoming Claude Code and AI coding tutorials on Medium***.

Claude Course Course

> **Every day I'm working hard on building the ultimate Claude Code course that demonstrates how to build workflows that coordinate multiple agents for complex development tasks. It's due for release soon.**

It will take what you have learned from this article to the next level of complete automation.

New features are added to Claude Code daily, and keeping up is tough.

The course explores subagents, hooks, advanced workflows, and productivity techniques that many developers may not discover.

Once you join, you'll receive all the updates as new features are rolled out.

This course will cover:

- *Advanced subagent patterns and workflows*
- *Production-ready hook configurations*
- *MCP server integrations for external tools*
- *Team collaboration strategies*
- *Enterprise deployment patterns*

- *Real-world case studies from my consulting work*

If you're interested in getting notified when the Claude Code course launches**, click here to join the early access list →**

(*Currently, I have 2000+ already signed-up developers)*

> ***I'll share exclusive previews, early access pricing, and bonus materials with people on the list.***