

## Use Claude Code with Ollama Models without Anthropic API Key

Fahd Mirza 53K subscribers Join Subscribe

2,913 views Jan 19, 2026 #claudecode  
This video locally installs Claude Code with Ollama models without any api key.

Learn to use Claude code with Ollama models locally, without an Anthropic API key. This tutorial demonstrates installation and setup, including environment variable configuration. The process involves using Ollama's messages API and explores tool calls.

The screenshot shows the Claude website with a dark theme. A central heading says "Built for > programmers". Below it, a subtext reads: "Work with Claude directly in your codebase. Build, debug, and ship from your terminal, IDE, Slack, or the web. Describe what you need, and Claude handles the rest." There is a "curl -fsSL https://claude.ai/install.sh | bash" command button. At the bottom, there are logos for StubHub, Brex, Intercom, and Notion.

Get Claude Code curl -fsSL https://claude.ai/install.sh | bash Or read the documentation

GitHub Discord Docs Cloud search models

Cloud Embedding Vision Tools Thinking Popular

**nemotron-3-nano**  
Nemotron 3 Nano - A new Standard for Efficient, Open, and Intelligent Agentic Models  
cloud 30b  
117.4K Pulls 6 Tags Updated 1 month ago

**functiongemma**  
FunctionGemma is a specialized version of Google's Gemma 3.270M model fine-tuned explicitly for function calling.  
tools 270m  
41.9K Pulls 4 Tags Updated 1 month ago

```
Ubuntu@0138-ict-prxmx50045:~$ ollama list
NAME          ID      SIZE    MODIFIED
qwen3-coder:latest  06c1097efce0  18 GB   2 minutes ago
gpt-oss:20b     17052f91a42e  13 GB   7 weeks ago
Ubuntu@0138-ict-prxmx50045:~$ ollama --version
ollama version is 0.14.2
Ubuntu@0138-ict-prxmx50045:~$ ollama pull
```

Make sure the model supports tool call or function use and have a context window of 64k or greater

```
Ubuntu@0138-ict-prxmx50045:~$ curl -fsSL https://claude.ai/install.sh | bash
Setting up Claude Code...
Installing Claude Code native build latest...
```

Install Claude Code using the command above

```
Ubuntu@0138-ict-prxmx50045:~$ curl -fsSL https://claude.ai/install.sh | bash
Setting up Claude Code...

✓ Claude Code successfully installed!

Version: 2.1.12

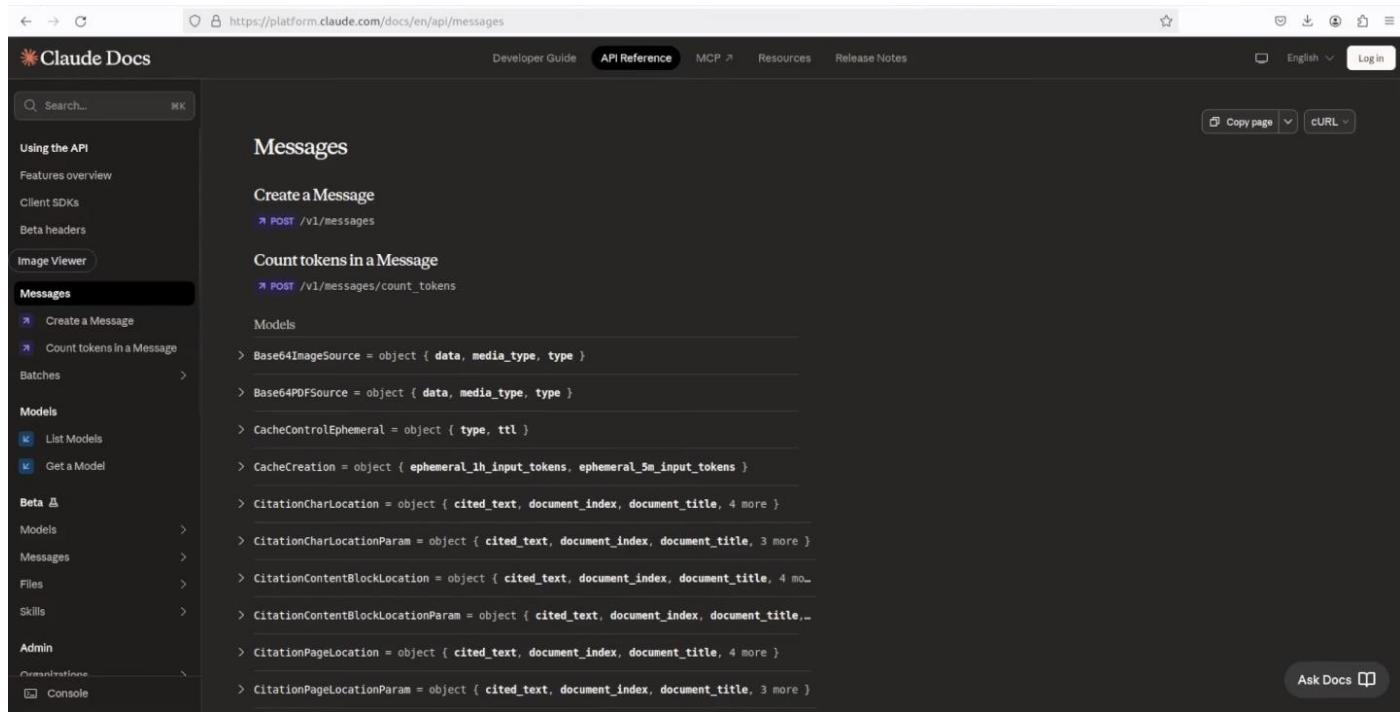
Location: ~/.local/bin/claude

Next: Run claude --help to get started

✓ Installation complete!
Ubuntu@0138-ict-prxmx50045:~$
```

```
Ubuntu@0138-ict-prxmx50045:~$ export ANTHROPIC_AUTH_TOKEN=oollama
export ANTHROPIC_BASE_URL=http://localhost:11434
Ubuntu@0138-ict-prxmx50045:~$
```

Set the env variables as above



The screenshot shows the Claude Docs API Reference website at <https://platform.claude.com/docs/en/api/messages>. The left sidebar has a 'Messages' section selected, which includes links for 'Create a Message' and 'Count tokens in a Message'. The main content area displays the 'Messages' API documentation, starting with the 'Create a Message' endpoint:

**Create a Message**  
↗ POST /v1/messages

**Count tokens in a Message**  
↗ POST /v1/messages/count\_tokens

Below these, there is a detailed description of the 'Models' object, listing various subclasses such as Base64ImageSource, Base64PDFSource, CacheControlEphemeral, CacheCreation, CitationCharLocation, CitationCharLocationParam, CitationContentBlockLocation, CitationContentBlockLocationParam, CitationPageLocation, and CitationPageLocationParam, each with their respective properties and descriptions.

This is because Ollama now supports the Messages API from Anthropic

```
Ubuntu@0138-ict-prxmx50045:~/mycode/testcode$ claude --model gpt-oss:20b

Do you trust the files in this folder?
/home/Ubuntu/mycode/testcode

Claude Code may read, write, or execute files contained in this directory. This can pose security
risks, so only use files from trusted sources.

Learn more
> 1. Yes, proceed
2. No, exit

Enter to confirm · Esc to cancel
```

You can now run Claude Code with your Ollama models as above, click Yes to proceed

The screenshot shows the Claude Code application window. On the left, there's a terminal-like interface with the following text:

```
Claude Code v2.1.12
Welcome back!
gpt-oss:20b · API Usage Billing
~/mycode/testcode

/model to try Opus 4.5
> try "fix typecheck errors"
? for shortcuts
```

The screenshot shows the Claude Code application window. On the left, there's a terminal-like interface with the following text:

```
Welcome back!
gpt-oss:20b · API Usage Billing
~/mycode/testcode

/model to try Opus 4.5
> write me a hello world program
* Symbioting... (ctrl+c to interrupt)

? for shortcuts
```

```
* Claude Code
```

/model to try Opus 4.5

write me a hello world program

- We should not mention todo list. Just give the program. Here's a simple "Hello, World!" program in a few common languages:

**Python**

```
print("Hello, World!")
```

**JavaScript (Node.js)**console.log("Hello, World!");

**Java**public class HelloWorld {  
 public static void main(String[] args) {  
 System.out.println("Hello, World!");  
 }  
}

```
File Edit Selection View Go Run Terminal Help
```

app.py - cco - Visual Studio Code

EXPLORER    ...    app.py    x

cco  
  app.py

```
1 import anthropic
2
3 client = anthropic.Anthropic(
4     base_url='http://localhost:11434',
5     api_key='ollama', # required but ignored
6 )
7
8 message = client.messages.create(
9     model='qwen3-coder',
10    messages=[
11        {'role': 'user', 'content': 'Write a function to check if a number is prime'}
12    ]
13 )
14 print(message.content[0].text)
```

You can also use Claude Code in your Python API code as above

```
Ubuntu@0138-ict-prxmx50045:~/mycode/cco$ pip install anthropic
Defaulting to user installation because normal site-packages is not writeable
Collecting anthropic
  Downloading anthropic-0.76.0-py3-none-any.whl (390 kB)
Collecting docstring-parser<1,>=0.15
  Downloading docstring_parser-0.17.0-py3-none-any.whl (36 kB)
Collecting jitter<1,>=0.4.0
  Downloading jitter-0.12.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (364 kB)
Collecting httpx<1,>=0.25.0
  Downloading httpx-0.28.1-py3-none-any.whl (73 kB)
Collecting anyio<5,>=3.5.0
  Downloading anyio-4.12.1-py3-none-any.whl (113 kB)
```

```
Ubuntu@0138-ict-prxmx50045:~/mycode/cco$ python3 app.py
Here are several implementations of a prime checking function in Python:

## Basic Implementation

```python
def is_prime(n):
    """
    Check if a number is prime.

    Args:
        n (int): The number to check

    Returns:
        bool: True if the number is prime, False otherwise
    """
    # Handle edge cases
    if n < 2:
        return False
    if n == 2:
        return True
    if n % 2 == 0:
```

```

1 is not prime
2 is prime
3 is prime
4 is not prime
5 is prime
17 is prime
25 is not prime
29 is prime
100 is not prime
101 is prime
```
## Key Points:
1. **Basic approach**: Check divisibility up to  $\sqrt{n}$ 
2. **Optimization**: Skip even numbers after 2
3. **Further optimization**: Check only numbers of the form  $6k \pm 1$ 
4. **Edge cases**: Handle numbers less than 2 appropriately

The first implementation (`is_prime`) is recommended for most use cases as it's efficient and easy to understand. The optimized version is faster for very large numbers but more complex to read.
Ubuntu@0138-ict-prxmx50045:~/mycode/cco$ 
```

```

File Edit Selection View Go Run Terminal Help
EXPLORER ... app.py x
app.py
1 import anthropic
2
3 client = anthropic.Anthropic(
4     base_url='http://localhost:11434',
5     api_key='ollama', # dummy - ignored by Ollama
6 )
7
8 message = client.messages.create(
9     model='qwen3-coder',
10    max_tokens=1024,
11    tools=[
12        {
13            'name': 'get_weather',
14            'description': 'Get the current weather in a location',
15            'input_schema': {
16                'type': 'object',
17                'properties': {
18                    'location': {
19                        'type': 'string',
20                        'description': 'The city and state, e.g. San Francisco, CA'
21                    }
22                },
23            },
24        }
25    ],
26    messages=[{'role': 'user', 'content': "What's the weather in San Francisco?"}]
27
28 )
```

Next, let us see a tool call example using Claude Code via Anthropic SDK call with the Ollama model

```

File Edit Selection View Go Run Terminal Help
EXPLORER ... app.py x
app.py
8 message = client.messages.create(
9     tools=[
10        {
11            'input_schema': {
12                'type': 'object',
13                'properties': {
14                    'location': {
15                        'type': 'string',
16                        'description': 'The city and state, e.g. San Francisco, CA'
17                    }
18                },
19            },
20            'required': ['location']
21        }
22    ],
23    messages=[{'role': 'user', 'content': "What's the weather in San Francisco?"}]
24
25 )
26
27 # Inspect the response blocks
28 for block in message.content:
29     if block.type == 'tool_use':
30         print(f'Tool: {block.name}')
31         print(f'Input: {block.input}')
32     elif block.type == 'text':
33         print('Text:', block.text)
```

```
Ubuntu@0138-ict-prxmx50045:~/mycode/cco$ python3 app.py
Tool: get weather
Input: {'location': 'San Francisco, CA'}
Ubuntu@0138-ict-prxmx50045:~/mycode/cco$
```

<https://docs.ollama.com/api/anthropic-compatibility>

**Endpoints**

**/v1/messages**

**Supported features**

- Messages
- Streaming
- System prompts
- Multi-turn conversations
- Vision (images)
- Tools (function calling)
- Tool results
- Thinking/extended thinking

**Supported request fields**

- model
- max\_tokens
- messages
  - Text\_content
  - Image\_content (base64)
  - Array of content blocks
  - tool\_use\_blocks
  - tool\_result\_blocks

You can do a lot more than using the Messages API here, you can do multi-turn conversations, streaming, system prompts, extended thinking, vision or image inputs. Look [docs.ollama.com](https://docs.ollama.com) for more information.

<https://docs.ollama.com/api/anthropic-compatibility>

**Endpoints**

**/v1/messages**

**Supported features**

- stop\_reason (end\_turn, max\_tokens, tool\_use)
- usage (input\_tokens, output\_tokens)

**Streaming events**

- message\_start
- content\_block\_start
- content\_block\_delta (text\_delta, input\_json\_delta, thinking\_delta)
- content\_block\_stop
- message\_delta
- message\_stop
- ping
- error

**Models**

Ollama supports both local and cloud models.

**Local models**

Pull a local model before use:

```
ollama pull qwen3-coder
```

Recommended local models:

- qwen3-coder - Excellent for coding tasks