

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



Fahd Mirza  
53K subscribers

Join

Subscribe

170



Share

Ask

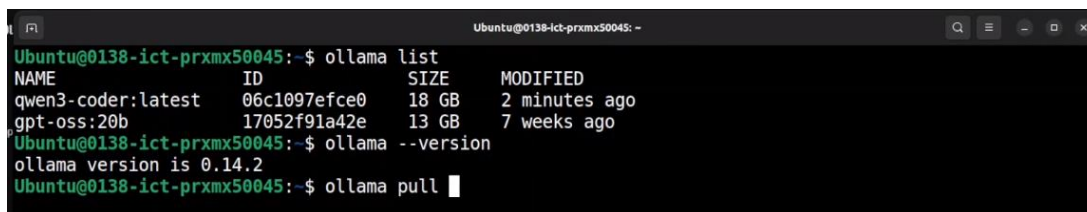
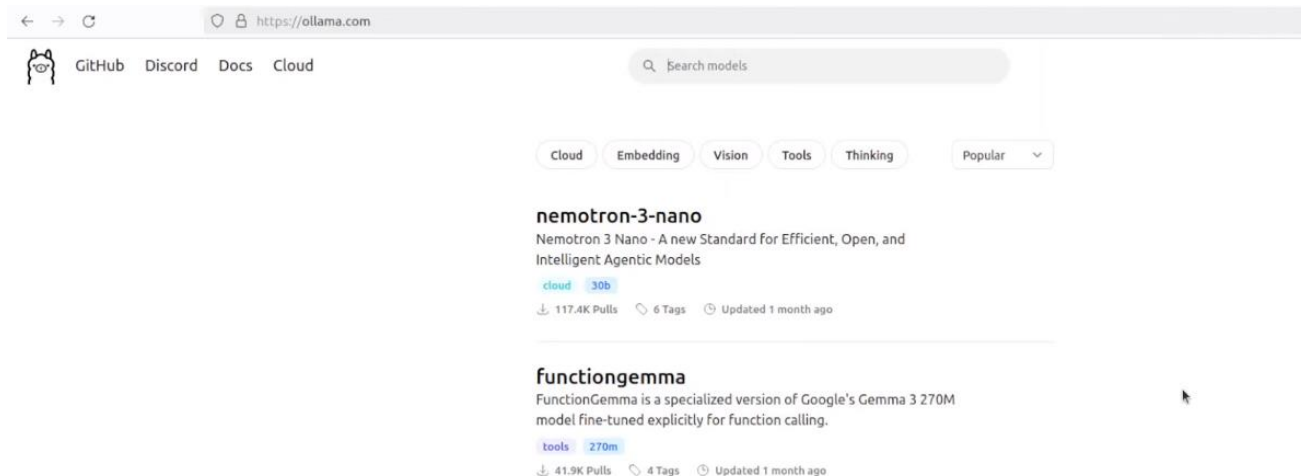
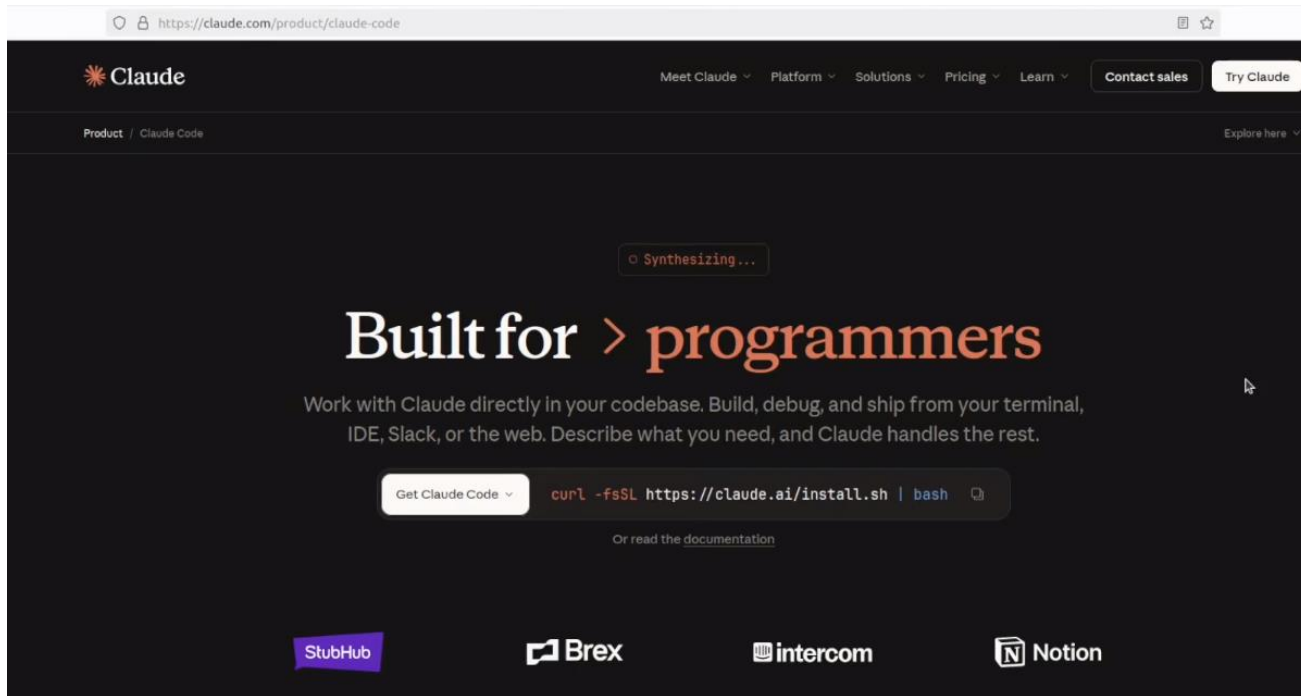
Save



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.



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

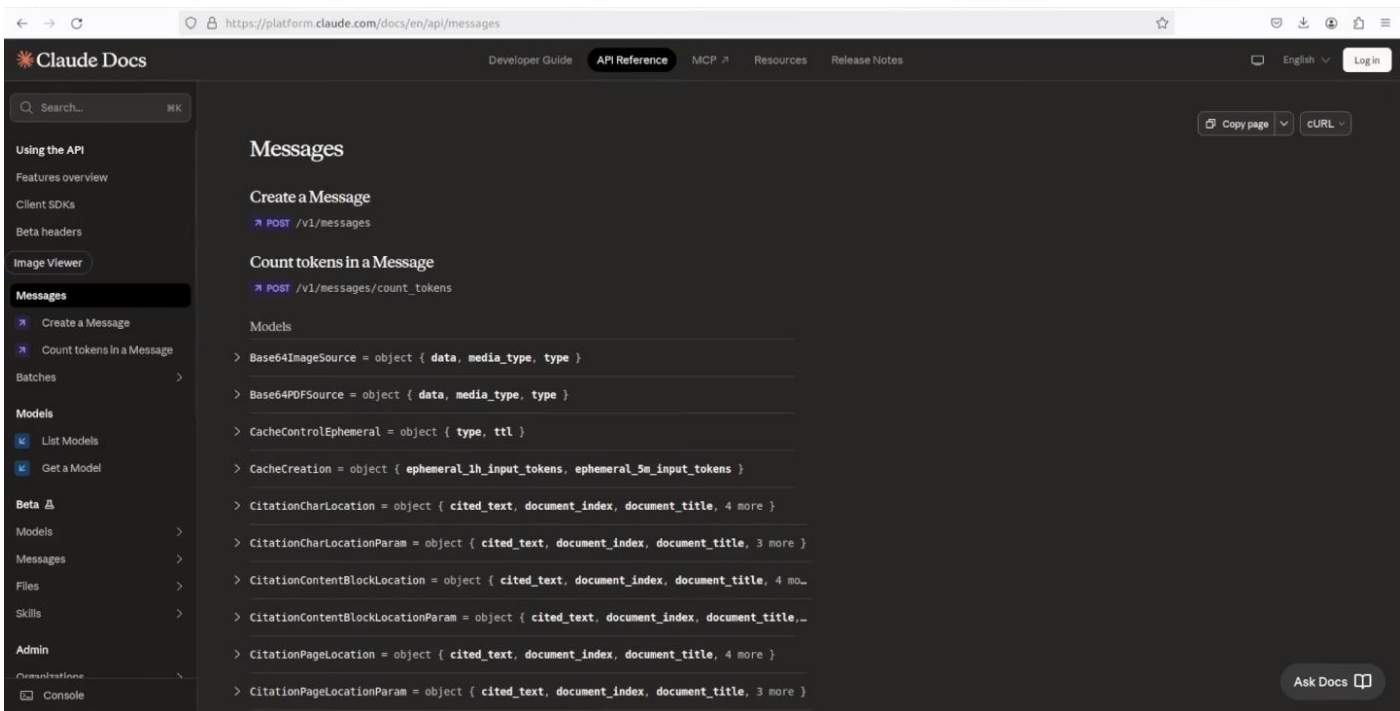
Next: Run claude --help to get started

✓ Installation complete!

Ubuntu@0138-ict-prxmx50045:~$
```

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

Set the env variables as above



The screenshot shows the Claude Docs website at the URL <https://platform.claude.com/docs/en/api/messages>. The page is titled "Messages" and contains the following sections:

- Create a Message**
  - POST /v1/messages
- Count tokens in a Message**
  - POST /v1/messages/count\_tokens
- Models**
  - Base64ImageSource = object { data, media\_type, type }
  - Base64PDFSource = object { data, media\_type, type }
  - CacheControlEphemeral = object { type, ttl }
  - CacheCreation = object { ephemeral\_1h\_input\_tokens, ephemeral\_5m\_input\_tokens }
  - CitationCharLocation = object { cited\_text, document\_index, document\_title, 4 more }
  - CitationCharLocationParam = object { cited\_text, document\_index, document\_title, 3 more }
  - CitationContentBlockLocation = object { cited\_text, document\_index, document\_title, 4 more }
  - CitationContentBlockLocationParam = object { cited\_text, document\_index, document\_title, 3 more }
  - CitationPageLocation = object { cited\_text, document\_index, document\_title, 4 more }
  - CitationPageLocationParam = object { cited\_text, document\_index, document\_title, 3 more }

The left sidebar shows the navigation menu with categories: Using the API, Features overview, Client SDKs, Beta headers, Image Viewer, Messages (selected), Batches, Models, Beta, Models, Messages, Files, Skills, Admin, and Console. The right sidebar has buttons for "Copy page" and "cURL".

This is because Ollama now supports the Messages API from Anthropic

```
Ubuntu@0138-ict-prxm50045: ~/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

```
* Claude Code

Claude Code v2.1.12

Welcome back!



gpt-oss:20b · API Usage Billing
~/mycode/testcode

Tips for getting started
Run /init to create a CLAUDE.md file with instructions for CL...

Recent activity
No recent activity

/model to try Opus 4.5


> Try "fix typecheck errors"

? for shortcuts
```

```
* Claude Code

Claude Code v2.1.12

Welcome back!



gpt-oss:20b · API Usage Billing
~/mycode/testcode

Tips for getting started
Run /init to create a CLAUDE.md file with instructions for CL...

Recent activity
No recent activity

/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!");
    }
}
```

```
 app.py - cco - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER
  cco
    app.py

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/ccco

Ubuntu@0138-ict-prxmx50045:~/mycode/ccco$ 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)
    390.3/390.3 KB 5.4 MB/s eta 0:00:00
Collecting docstring-parser<1,>=0.15
  Downloading docstring_parser-0.17.0-py3-none-any.whl (36 kB)
Collecting jiter<1,>=0.4.0
  Downloading jiter-0.12.0-cp310-cp310-manylinux_2_17_x86_64_manylinux2014_x86_64.whl (364 kB)
    364.4/364.4 KB 30.0 MB/s eta 0:00:00
Collecting httpx<1,>=0.25.0
  Downloading httpx-0.28.1-py3-none-any.whl (73 kB)
    73.5/73.5 KB 17.5 MB/s eta 0:00:00
Collecting anyio<5,>=3.5.0
  Downloading anyio-4.12.1-py3-none-any.whl (113 kB)
    0.0/113.6 KB ? eta -:-:--
```

```
 Ubuntu@0138-ict-prxmx50045: ~/mycode/ccco

Ubuntu@0138-ict-prxmx50045:~/mycode/ccco$ 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:
        return False
    for i in range(3, int(n**0.5) + 1, 2):
        if n % i == 0:
            return False
    return True

```

```

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/cc0$

```

```

app.py - cco - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER
  cco
    app.py

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                 'required': ['location']
24             }
25         }
26     ],
27     messages=[{'role': 'user', 'content': "What's the weather in San Francisco?"}]
28 )
29

```

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

```

app.py - cco - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER
  cco
    app.py

app.py
8  message = client.messages.create(
11     tools=[
12         {
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                 'required': ['location']
24             }
25         }
26     ],
27     messages=[{'role': 'user', 'content': "What's the weather in San Francisco?"}]
28 )
29
30 # Inspect the response blocks
31 for block in message.content:
32     if block.type == 'tool_use':
33         print(f'Tool: {block.name}')
34         print(f'Input: {block.input}')
35     elif block.type == 'text':
36         print('Text:', block.text)

```

