

Thinking



- [Discover a model's thinking controls](#)
 - [Enable thinking in API calls](#)
 - [Stream the reasoning trace](#)
-

Thinking-capable models emit a `thinking` field that separates their reasoning trace from the final answer.

Use this capability to audit model steps, animate the model *thinking* in a UI, or hide the trace entirely when you only need the final response.

See the [full list of thinking models](#).

Discover a model's thinking controls

Thinking controls vary by model. Use `/api/show` to discover the values a model supports and the value Ollama uses by default:

```
curl http://localhost:11434/api/show -d '{
  "model": "gpt-oss"
}'
```

The response includes a top-level `thinking` object:

```
{
  "thinking": {
    "values": ["low", "medium", "high"],
    "default": "medium"
  }
}
```

- `values` can contain booleans (`true` or `false`) for on/off controls. It can also contain model-defined strings for named levels.
- `default` is used when `think` is not set.
- `values: [false]` means the model does not support thinking.
- If `thinking` is omitted, the model has no thinking metadata. The model might conduct thinking based on its existing behavior.

Enable thinking in API calls

Set the `think` field on a chat or generate request:

- `true`: request thinking output.
- `false`: request no thinking output, if the model permits it.
- `null`: use the model default.
- A string: select a supported level from `thinking.values`. Use the exact value from `/api/show`. Numbers are not supported.

If the model resolves named levels from `/api/show` metadata, Ollama applies supported names exactly. Unsupported names use the model default.

The reasoning output and answer use separate fields. Chat returns `message.thinking` and `message.content`. Generate returns `thinking` and `response`.

```
```shell curl http://localhost:11434/api/chat -d '{ "model": "qwen3", "messages": [{ "role": "user",
"content": "How many letter r are in strawberry?" }], "think": true, "stream": false }' ```
python
from ollama import chat
```

```
response = chat(
 model='qwen3',
 messages=[{'role': 'user', 'content': 'How many letter r are in strawberry?'}],
 think=True,
 stream=False,
)

print('Thinking:\n', response.message.thinking)
print('Answer:\n', response.message.content)
```
```

```
```javascript import ollama from 'ollama'
```

```
const response = await ollama.chat({
 model: 'deepseek-r1',
 messages: [{ role: 'user', content: 'How many letter r are in strawberry?' }],
```

```

 think: true,
 stream: false,
 })

 console.log('Thinking:\n', response.message.thinking)
 console.log('Answer:\n', response.message.content)
 ...

```

# Stream the reasoning trace

Thinking streams interleave reasoning tokens before answer tokens. Detect the first `thinking` chunk to render a "thinking" section, then switch to the final reply once `message.content` arrives.

```

```python from ollama import chat

```

```

stream = chat(
    model='qwen3',
    messages=[{'role': 'user', 'content': 'What is 17 × 23?'}],
    think=True,
    stream=True,
)

in_thinking = False

for chunk in stream:
    if chunk.message.thinking and not in_thinking:
        in_thinking = True
        print('Thinking:\n', end='')

    if chunk.message.thinking:
        print(chunk.message.thinking, end='')
    elif chunk.message.content:
        if in_thinking:
            print('\n\nAnswer:\n', end='')
            in_thinking = False
        print(chunk.message.content, end='')

...

```

```

```javascript import ollama from 'ollama'

```

```

async function main() {
 const stream = await ollama.chat({
 model: 'qwen3',
 messages: [{ role: 'user', content: 'What is 17 × 23?' }],
 think: true,
 stream: true,
 })

 let inThinking = false

 for await (const chunk of stream) {
 if (chunk.message.thinking && !inThinking) {
 inThinking = true
 process.stdout.write('Thinking:\n')
 }

 if (chunk.message.thinking) {
 process.stdout.write(chunk.message.thinking)
 } else if (chunk.message.content) {
 if (inThinking) {
 process.stdout.write('\n\nAnswer:\n')
 inThinking = false
 }
 process.stdout.write(chunk.message.content)
 }
 }
}

main()
...

```

---

## #2

## 2026-10-31 13:16:51 UTC

## 2026-10-31 13:16:52 UTC