API

> Note: Ollama's API docs are moving to https://docs.ollama.com/api

Endpoints

Conventions

Model names

Model names follow a `model:tag` format, where `model` can have an optional namespace such as `example/model`. Some examples are `orca-mini:3b-q8_0` and `llama3:70b`. The tag is optional and, if not provided, will default to `latest`. The tag is used to identify a specific version.

Durations

All durations are returned in nanoseconds.

Streaming responses

Certain endpoints stream responses as JSON objects. Streaming can be disabled by providing `{"stream": false}` for these endpoints.

Generate a completion

```
POST /api/generate
```

Generate a response for a given prompt with a provided model. This is a streaming endpoint, so there will be a series of responses. The final response object will include statistics and additional data from the request.

Parameters

- `model`: (required) the model name
- `prompt`: the prompt to generate a response for
- `suffix`: the text after the model response
- `images`: (optional) a list of base64-encoded images (for multimodal models such as `llava`)
- `think`: (for thinking models) should the model think before responding? Can be a boolean or a thinking level (`"low"`, `"medium"`, `"high"`, or `"max"`).

Advanced parameters (optional):

- `format`: the format to return a response in. Format can be `json` or a JSON schema
- `options`: additional model parameters listed in the documentation for the Modelfile such as `temperature`
- `system`: system message to (overrides what is defined in the `Modelfile`)
- `template`: the prompt template to use (overrides what is defined in the `Modelfile`)

- `stream`: if `false` the response will be returned as a single response object, rather than a stream of objects
- `raw`: if `true` no formatting will be applied to the prompt. You may choose to use the `raw` parameter if you are specifying a full templated prompt in your request to the API
- `keep_alive`: controls how long the model will stay loaded into memory following the request (default: `5m`)
- `context` (deprecated): the context parameter returned from a previous request to `/generate`, this can be used to keep a short conversational memory

Structured outputs

Structured outputs are supported by providing a JSON schema in the `format` parameter. The model will generate a response that matches the schema. See the structured outputs example below.

JSON mode

Enable JSON mode by setting the `format` parameter to `json`. This will structure the response as a valid JSON object. See the JSON mode example below.

> Important
>
> It's important to instruct the model to use JSON in the `prompt`. Otherwise, the model may generate large amounts whitespace.

Examples

Generate request (Streaming)

Request

```
curl http://localhost:11434/api/generate -d '{
  "model": "llama3.2",
  "prompt": "Why is the sky blue?"
}'
```

Response

A stream of JSON objects is returned:

```
{
  "model": "llama3.2",
  "created_at": "2023-08-04T08:52:19.385406455-07:00",
  "response": "The",
  "done": false
}
```

The final response in the stream also includes additional data about the generation:

- `total_duration`: time spent generating the response
- `load_duration`: time spent in nanoseconds loading the model
- `prompt_eval_count`: number of tokens in the prompt
- `prompt_eval_duration`: time spent in nanoseconds evaluating the prompt
- `eval_count`: number of tokens in the response
- `eval_duration`: time in nanoseconds spent generating the response
- `context`: an encoding of the conversation used in this response, this can be sent in the next request to keep a conversational memory
- `response`: empty if the response was streamed, if not streamed, this will contain the full response

To calculate how fast the response is generated in tokens per second (token/s), divide `eval_count` / `eval_duration` * `10^9`.

```
{
  "model": "llama3.2",
  "created_at": "2023-08-04T19:22:45.499127Z",
  "response": "",
  "done": true,
  "context": [1, 2, 3],
  "total_duration": 10706818083,
  "load_duration": 6338219291,
  "prompt_eval_count": 26,
  "prompt_eval_duration": 130079000,
  "eval_count": 259,
  "eval_duration": 4232710000
}
```

Request (No streaming)

Request

A response can be received in one reply when streaming is off.

```
curl http://localhost:11434/api/generate -d '{
  "model": "llama3.2",
  "prompt": "Why is the sky blue?",
  "stream": false
}'
```

Response

If `stream` is set to `false`, the response will be a single JSON object:

```
{
  "model": "llama3.2",
  "created_at": "2023-08-04T19:22:45.499127Z",
  "response": "The sky is blue because it is the color of the sky.",
  "done": true,
  "context": [1, 2, 3],
  "total_duration": 5043500667,
  "load_duration": 5025959,
  "prompt_eval_count": 26,
  "prompt_eval_duration": 325953000,
  "eval_count": 290,
  "eval_duration": 4709213000
}
```

Request (with suffix)

Request

```
curl http://localhost:11434/api/generate -d '{
  "model": "codellama:code",
  "prompt": "def compute_gcd(a, b):",
  "suffix": "    return result",
  "options": {
    "temperature": 0
  },
  "stream": false
}'
```

Response

```
{
  "model": "codellama:code",
  "created_at": "2024-07-22T20:47:51.147561Z",
  "response": "\n  if a == 0:\n    return b\n  else:\n    return compute_gcd
```

```
  "done": true,
  "done_reason": "stop",
  "context": [...],
  "total_duration": 1162761250,
  "load_duration": 6683708,
  "prompt_eval_count": 17,
  "prompt_eval_duration": 201222000,
  "eval_count": 63,
  "eval_duration": 953997000
}
```

Request (Structured outputs)

Request

```
curl -X POST http://localhost:11434/api/generate -H "Content-Type: applicati
  "model": "llama3.1:8b",
  "prompt": "Ollama is 22 years old and is busy saving the world. Respond us
  "stream": false,
  "format": {
    "type": "object",
    "properties": {
      "age": {
        "type": "integer"
      },
      "available": {
        "type": "boolean"
      }
    },
    "required": [
      "age",
      "available"
    ]
  }
}'
```

Response

```
{
  "model": "llama3.1:8b",
  "created_at": "2024-12-06T00:48:09.983619Z",
  "response": "{\n  \"age\": 22,\n  \"available\": true\n}",
  "done": true,
  "done_reason": "stop",
```

```
  "context": [1, 2, 3],
  "total_duration": 1075509083,
  "load_duration": 567678166,
  "prompt_eval_count": 28,
  "prompt_eval_duration": 236000000,
  "eval_count": 16,
  "eval_duration": 269000000
}
```

Request (JSON mode)

> Important
>
> When `format` is set to `json`, the output will always be a well-formed JSON object. It's important to also instruct the model to respond in JSON.

Request

```
curl http://localhost:11434/api/generate -d '{
  "model": "llama3.2",
  "prompt": "What color is the sky at different times of the day? Respond us
  "format": "json",
  "stream": false
}'
```

Response

```
{
  "model": "llama3.2",
  "created_at": "2023-11-09T21:07:55.186497Z",
  "response": "{\n\"morning\": {\n\"color\": \"blue\"\n},\n\"noon\": {\n\"co
  "done": true,
  "context": [1, 2, 3],
  "total_duration": 4648158584,
  "load_duration": 4071084,
  "prompt_eval_count": 36,
  "prompt_eval_duration": 439038000,
  "eval_count": 180,
  "eval_duration": 4196918000
}
```

The value of `response` will be a string containing JSON similar to:

```
{
  "morning": {
    "color": "blue"
  },
  "noon": {
    "color": "blue-gray"
  },
  "afternoon": {
    "color": "warm gray"
  },
  "evening": {
    "color": "orange"
  }
}
```

Request (with images)

To submit images to multimodal models such as `llava` or `bakllava`, provide a list of base64-encoded `images`:

Request

```
curl http://localhost:11434/api/generate -d '{
  "model": "llava",
  "prompt":"What is in this picture?",
  "stream": false,
  "images": ["iVBORw0KGgoAAAANSUhEUgAAAG0AAABmCAYAAADBPx+VAAAACXBIWXMAAAsTA
}'
```

Response

```
{
  "model": "llava",
  "created_at": "2023-11-03T15:36:02.583064Z",
  "response": "A happy cartoon character, which is cute and cheerful.",
  "done": true,
  "context": [1, 2, 3],
  "total_duration": 2938432250,
  "load_duration": 2559292,
  "prompt_eval_count": 1,
  "prompt_eval_duration": 2195557000,
  "eval_count": 44,
```

```
  "eval_duration": 736432000
}
```

Request (Raw Mode)

In some cases, you may wish to bypass the templating system and provide a full prompt. In this case, you can use the `raw` parameter to disable templating. Also note that raw mode will not return a context.

Request

```
curl http://localhost:11434/api/generate -d '{
  "model": "mistral",
  "prompt": "[INST] why is the sky blue? [/INST]",
  "raw": true,
  "stream": false
}'
```

Request (Reproducible outputs)

For reproducible outputs, set `seed` to a number:

Request

```
curl http://localhost:11434/api/generate -d '{
  "model": "mistral",
  "prompt": "Why is the sky blue?",
  "options": {
    "seed": 123
  }
}'
```

Response

```
{
  "model": "mistral",
  "created_at": "2023-11-03T15:36:02.583064Z",
  "response": " The sky appears blue because of a phenomenon called Rayleigh
  "done": true,
  "total_duration": 8493852375,
  "load_duration": 6589624375,
  "prompt_eval_count": 14,
  "prompt_eval_duration": 119039000,
  "eval_count": 110,
```

```
  "eval_duration": 1779061000
}
```

Generate request (With options)

If you want to set custom options for the model at runtime rather than in the Modelfile, you can do so with the `options` parameter. This example sets every available option, but you can set any of them individually and omit the ones you do not want to override.

Request

```
curl http://localhost:11434/api/generate -d '{
  "model": "llama3.2",
  "prompt": "Why is the sky blue?",
  "stream": false,
  "options": {
    "num_keep": 5,
    "seed": 42,
    "num_predict": 100,
    "draft_num_predict": 4,
    "top_k": 20,
    "top_p": 0.9,
    "min_p": 0.0,
    "typical_p": 0.7,
    "repeat_last_n": 33,
    "temperature": 0.8,
    "repeat_penalty": 1.2,
    "presence_penalty": 1.5,
    "frequency_penalty": 1.0,
    "penalize_newline": true,
    "stop": ["\n", "user:"],
    "numa": false,
    "num_ctx": 1024,
    "num_batch": 2,
    "num_gpu": 1,
    "main_gpu": 0,
    "use_mmap": true,
    "num_thread": 8
  }
}'
```

Response

```
{
  "model": "llama3.2",
```

```
  "created_at": "2023-08-04T19:22:45.499127Z",
  "response": "The sky is blue because it is the color of the sky.",
  "done": true,
  "context": [1, 2, 3],
  "total_duration": 4935886791,
  "load_duration": 534986708,
  "prompt_eval_count": 26,
  "prompt_eval_duration": 107345000,
  "eval_count": 237,
  "eval_duration": 4289432000
}
```

Load a model

If an empty prompt is provided, the model will be loaded into memory.

Request

```
curl http://localhost:11434/api/generate -d '{
  "model": "llama3.2"
}'
```

Response

A single JSON object is returned:

```
{
  "model": "llama3.2",
  "created_at": "2023-12-18T19:52:07.071755Z",
  "response": "",
  "done": true
}
```

Unload a model

If an empty prompt is provided and the `keep_alive` parameter is set to `0`, a model will be unloaded from memory.

Request

```
curl http://localhost:11434/api/generate -d '{
  "model": "llama3.2",
```

```
  "keep_alive": 0
}'
```

Response

A single JSON object is returned:

```
{
  "model": "llama3.2",
  "created_at": "2024-09-12T03:54:03.516566Z",
  "response": "",
  "done": true,
  "done_reason": "unload"
}
```

Generate a chat completion

```
POST /api/chat
```

Generate the next message in a chat with a provided model. This is a streaming endpoint, so there will be a series of responses. Streaming can be disabled using `"stream": false`. The final response object will include statistics and additional data from the request.

Parameters

- `model`: (required) the model name
- `messages`: the messages of the chat, this can be used to keep a chat memory
- `tools`: list of tools in JSON for the model to use if supported
- `think`: (for thinking models) should the model think before responding? Can be a boolean or a thinking level (`"low"`, `"medium"`, `"high"`, or `"max"`).

The `message` object has the following fields:

- `role`: the role of the message, either `system`, `user`, `assistant`, or `tool`
- `content`: the content of the message
- `thinking`: (for thinking models) the model's thinking process
- `images` (optional): a list of images to include in the message (for multimodal models such as `llava`)
- `tool_calls` (optional): a list of tools in JSON that the model wants to use

- `tool_name` (optional): add the name of the tool that was executed to inform the model of the result

Advanced parameters (optional):

- `format`: the format to return a response in. Format can be `json` or a JSON schema.
- `options`: additional model parameters listed in the documentation for the Modelfile such as `temperature`
- `stream`: if `false` the response will be returned as a single response object, rather than a stream of objects
- `keep_alive`: controls how long the model will stay loaded into memory following the request (default: `5m`)

Tool calling

Tool calling is supported by providing a list of tools in the `tools` parameter. The model will generate a response that includes a list of tool calls. See the Chat request (Streaming with tools) example below.

Models can also explain the result of the tool call in the response. See the Chat request (With history, with tools) example below.

See models with tool calling capabilities.

Structured outputs

Structured outputs are supported by providing a JSON schema in the `format` parameter. The model will generate a response that matches the schema. See the Chat request (Structured outputs) example below.

Examples

Chat request (Streaming)

Request

Send a chat message with a streaming response.

```
curl http://localhost:11434/api/chat -d '{
  "model": "llama3.2",
  "messages": [
    {
      "role": "user",
```

```
      "content": "why is the sky blue?"
    }
  ]
}'
```

Response

A stream of JSON objects is returned:

```
{
  "model": "llama3.2",
  "created_at": "2023-08-04T08:52:19.385406455-07:00",
  "message": {
    "role": "assistant",
    "content": "The",
    "images": null
  },
  "done": false
}
```

Final response:

```
{
  "model": "llama3.2",
  "created_at": "2023-08-04T19:22:45.499127Z",
  "message": {
    "role": "assistant",
    "content": ""
  },
  "done": true,
  "total_duration": 4883583458,
  "load_duration": 1334875,
  "prompt_eval_count": 26,
  "prompt_eval_duration": 342546000,
  "eval_count": 282,
  "eval_duration": 4535599000
}
```

Chat request (Streaming with tools)

Request

```
curl http://localhost:11434/api/chat -d '{
  "model": "llama3.2",
  "messages": [
```

```
    {
      "role": "user",
      "content": "what is the weather in tokyo?"
    }
  ],
  "tools": [
    {
      "type": "function",
      "function": {
        "name": "get_weather",
        "description": "Get the weather in a given city",
        "parameters": {
          "type": "object",
          "properties": {
            "city": {
              "type": "string",
              "description": "The city to get the weather for"
            }
          },
          "required": ["city"]
        }
      }
    }
  ],
  "stream": true
}'
```

Response

A stream of JSON objects is returned:

```
{
  "model": "llama3.2",
  "created_at": "2025-07-07T20:22:19.184789Z",
  "message": {
    "role": "assistant",
    "content": "",
    "tool_calls": [
      {
        "function": {
          "name": "get_weather",
          "arguments": {
            "city": "Tokyo"
          }
        }
      }
    ]
  },
```

```
  "done": false
}
```

Final response:

```
{
  "model": "llama3.2",
  "created_at": "2025-07-07T20:22:19.19314Z",
  "message": {
    "role": "assistant",
    "content": ""
  },
  "done_reason": "stop",
  "done": true,
  "total_duration": 182242375,
  "load_duration": 41295167,
  "prompt_eval_count": 169,
  "prompt_eval_duration": 24573166,
  "eval_count": 15,
  "eval_duration": 115959084
}
```

Chat request (No streaming)

Request

```
curl http://localhost:11434/api/chat -d '{
  "model": "llama3.2",
  "messages": [
    {
      "role": "user",
      "content": "why is the sky blue?"
    }
  ],
  "stream": false
}'
```

Response

```
{
  "model": "llama3.2",
  "created_at": "2023-12-12T14:13:43.416799Z",
  "message": {
    "role": "assistant",
    "content": "Hello! How are you today?"
```

```
  },
  "done": true,
  "total_duration": 5191566416,
  "load_duration": 2154458,
  "prompt_eval_count": 26,
  "prompt_eval_duration": 383809000,
  "eval_count": 298,
  "eval_duration": 4799921000
}
```

Chat request (No streaming, with tools)

Request

```
curl http://localhost:11434/api/chat -d '{
  "model": "llama3.2",
  "messages": [
    {
      "role": "user",
      "content": "what is the weather in tokyo?"
    }
  ],
  "tools": [
    {
      "type": "function",
      "function": {
        "name": "get_weather",
        "description": "Get the weather in a given city",
        "parameters": {
          "type": "object",
          "properties": {
            "city": {
              "type": "string",
              "description": "The city to get the weather for"
            }
          },
          "required": ["city"]
        }
      }
    }
  ],
  "stream": false
}'
```

Response

```
{
  "model": "llama3.2",
  "created_at": "2025-07-07T20:32:53.844124Z",
  "message": {
    "role": "assistant",
    "content": "",
    "tool_calls": [
      {
        "function": {
          "name": "get_weather",
          "arguments": {
            "city": "Tokyo"
          }
        }
      }
    ]
  },
  "done_reason": "stop",
  "done": true,
  "total_duration": 3244883583,
  "load_duration": 2969184542,
  "prompt_eval_count": 169,
  "prompt_eval_duration": 141656333,
  "eval_count": 18,
  "eval_duration": 133293625
}
```


Chat request (Structured outputs)

Request

```
curl -X POST http://localhost:11434/api/chat -H "Content-Type: application/
  "model": "llama3.1",
  "messages": [{"role": "user", "content": "Ollama is 22 years old and busy
  "stream": false,
  "format": {
    "type": "object",
    "properties": {
      "age": {
        "type": "integer"
      },
      "available": {
        "type": "boolean"
      }
    },
    "required": [
      "age",
```

```
        "available"
      ]
    },
    "options": {
      "temperature": 0
    }
}'
```

Response

```
{
  "model": "llama3.1",
  "created_at": "2024-12-06T00:46:58.265747Z",
  "message": {
    "role": "assistant",
    "content": "{\"age\": 22, \"available\": false}"
  },
  "done_reason": "stop",
  "done": true,
  "total_duration": 2254970291,
  "load_duration": 574751416,
  "prompt_eval_count": 34,
  "prompt_eval_duration": 1502000000,
  "eval_count": 12,
  "eval_duration": 175000000
}
```

Chat request (With History)

Send a chat message with a conversation history. You can use this same approach to start the conversation using multi-shot or chain-of-thought prompting.

Request

```
curl http://localhost:11434/api/chat -d '{
  "model": "llama3.2",
  "messages": [
    {
      "role": "user",
      "content": "why is the sky blue?"
    },
    {
      "role": "assistant",
      "content": "due to rayleigh scattering."
    },
```

```
    {
      "role": "user",
      "content": "how is that different than mie scattering?"
    }
  ]
}'
```

Response

A stream of JSON objects is returned:

```
{
  "model": "llama3.2",
  "created_at": "2023-08-04T08:52:19.385406455-07:00",
  "message": {
    "role": "assistant",
    "content": "The"
  },
  "done": false
}
```

Final response:

```
{
  "model": "llama3.2",
  "created_at": "2023-08-04T19:22:45.499127Z",
  "done": true,
  "total_duration": 8113331500,
  "load_duration": 6396458,
  "prompt_eval_count": 61,
  "prompt_eval_duration": 398801000,
  "eval_count": 468,
  "eval_duration": 7701267000
}
```

Chat request (With history, with tools)

Request

```
curl http://localhost:11434/api/chat -d '{
  "model": "llama3.2",
  "messages": [
    {
      "role": "user",
      "content": "what is the weather in Toronto?"
```

```
    },
    // the message from the model appended to history
    {
      "role": "assistant",
      "content": "",
      "tool_calls": [
        {
          "function": {
            "name": "get_weather",
            "arguments": {
              "city": "Toronto"
            }
          }
        }
      ]
    },
    // the tool call result appended to history
    {
      "role": "tool",
      "content": "11 degrees celsius",
      "tool_name": "get_weather"
    }
  ],
  "stream": false,
  "tools": [
    {
      "type": "function",
      "function": {
        "name": "get_weather",
        "description": "Get the weather in a given city",
        "parameters": {
          "type": "object",
          "properties": {
            "city": {
              "type": "string",
              "description": "The city to get the weather for"
            }
          },
          "required": ["city"]
        }
      }
    }
  ]
}'
```

Response

```
{
  "model": "llama3.2",
  "created_at": "2025-07-07T20:43:37.688511Z",
  "message": {
    "role": "assistant",
    "content": "The current temperature in Toronto is 11°C."
  },
  "done_reason": "stop",
  "done": true,
  "total_duration": 890771750,
  "load_duration": 707634750,
  "prompt_eval_count": 94,
  "prompt_eval_duration": 91703208,
  "eval_count": 11,
  "eval_duration": 90282125
}
```


Chat request (with images)

Request

Send a chat message with images. The images should be provided as an array, with the individual images encoded in Base64.


```
curl http://localhost:11434/api/chat -d '{
  "model": "llava",
  "messages": [
    {
      "role": "user",
      "content": "what is in this image?",
      "images": ["iVBORw0KGgoAAAANSUhEUgAAAG0AAABmCAYAAADBPx+VAAAACXBIWXMAA
    }
  ]
}'
```


Response


```
{
  "model": "llava",
  "created_at": "2023-12-13T22:42:50.203334Z",
  "message": {
    "role": "assistant",
    "content": " The image features a cute, little pig with an angry facial
    "images": null
```

```
  },
  "done": true,
  "total_duration": 1668506709,
  "load_duration": 1986209,
  "prompt_eval_count": 26,
  "prompt_eval_duration": 359682000,
  "eval_count": 83,
  "eval_duration": 1303285000
}
```

Chat request (Reproducible outputs)

Request

```
curl http://localhost:11434/api/chat -d '{
  "model": "llama3.2",
  "messages": [
    {
      "role": "user",
      "content": "Hello!"
    }
  ],
  "options": {
    "seed": 101,
    "temperature": 0
  }
}'
```

Response

```
{
  "model": "llama3.2",
  "created_at": "2023-12-12T14:13:43.416799Z",
  "message": {
    "role": "assistant",
    "content": "Hello! How are you today?"
  },
  "done": true,
  "total_duration": 5191566416,
  "load_duration": 2154458,
  "prompt_eval_count": 26,
  "prompt_eval_duration": 383809000,
  "eval_count": 298,
```

```
  "eval_duration": 4799921000
}
```

Chat request (with tools)

Request

```
curl http://localhost:11434/api/chat -d '{
  "model": "llama3.2",
  "messages": [
    {
      "role": "user",
      "content": "What is the weather today in Paris?"
    }
  ],
  "stream": false,
  "tools": [
    {
      "type": "function",
      "function": {
        "name": "get_current_weather",
        "description": "Get the current weather for a location",
        "parameters": {
          "type": "object",
          "properties": {
            "location": {
              "type": "string",
              "description": "The location to get the weather for, e.g. San
            },
            "format": {
              "type": "string",
              "description": "The format to return the weather in, e.g. 'cel
              "enum": ["celsius", "fahrenheit"]
            }
          },
          "required": ["location", "format"]
        }
      }
    }
  ]
}'
```

Response

```
{
  "model": "llama3.2",
  "created_at": "2024-07-22T20:33:28.123648Z",
  "message": {
    "role": "assistant",
    "content": "",
    "tool_calls": [
      {
        "function": {
          "name": "get_current_weather",
          "arguments": {
            "format": "celsius",
            "location": "Paris, FR"
          }
        }
      }
    ]
  },
  "done_reason": "stop",
  "done": true,
  "total_duration": 885095291,
  "load_duration": 3753500,
  "prompt_eval_count": 122,
  "prompt_eval_duration": 328493000,
  "eval_count": 33,
  "eval_duration": 552222000
}
```


Load a model

If the messages array is empty, the model will be loaded into memory.

Request

```
curl http://localhost:11434/api/chat -d '{
  "model": "llama3.2",
  "messages": []
}'
```

Response


```
{
  "model": "llama3.2",
  "created_at": "2024-09-12T21:17:29.110811Z",
  "message": {
```

```
    "role": "assistant",
    "content": ""
  },
  "done_reason": "load",
  "done": true
}
```

Unload a model

If the messages array is empty and the `keep_alive` parameter is set to `0`, a model will be unloaded from memory.

Request

```
curl http://localhost:11434/api/chat -d '{
  "model": "llama3.2",
  "messages": [],
  "keep_alive": 0
}'
```

Response

A single JSON object is returned:

```
{
  "model": "llama3.2",
  "created_at": "2024-09-12T21:33:17.547535Z",
  "message": {
    "role": "assistant",
    "content": ""
  },
  "done_reason": "unload",
  "done": true
}
```

Create a Model

```
POST /api/create
```

Create a model from:

- another model;

- a safetensors directory; or
- a GGUF file.

If you are creating a model from a safetensors directory or from a GGUF file, you must push a blob for each of the files and then use the file name and SHA256 digest associated with each blob in the `files` field.

Parameters

- `model`: name of the model to create
- `from`: (optional) name of an existing model to create the new model from
- `files`: (optional) a dictionary of file names to SHA256 digests of blobs to create the model from
- `adapters`: (optional) a dictionary of file names to SHA256 digests of blobs for LORA adapters
- `template`: (optional) the prompt template for the model
- `renderer`: (optional) the name of the renderer for the model
- `parser`: (optional) the name of the parser for the model
- `license`: (optional) a string or list of strings containing the license or licenses for the model
- `system`: (optional) a string containing the system prompt for the model
- `parameters`: (optional) a dictionary of parameters for the model (see Modelfile for a list of parameters)
- `messages`: (optional) a list of message objects used to create a conversation
- `stream`: (optional) if `false` the response will be returned as a single response object, rather than a stream of objects
- `quantize` (optional): quantize a non-quantized (e.g. float16) model

Quantization types

Type	Recommended
q4_K_M	*
q4_K_S	
q8_0	*

Examples

Create a new model

Create a new model from an existing model.

Request

```
curl http://localhost:11434/api/create -d '{
  "model": "mario",
  "from": "llama3.2",
  "system": "You are Mario from Super Mario Bros."
}'
```

Response

A stream of JSON objects is returned:

```
{"status":"reading model metadata"}
{"status":"creating system layer"}
{"status":"using already created layer sha256:22f7f8ef5f4c791c1b03d7eb414399
{"status":"using already created layer sha256:8c17c2ebb0ea011be9981cc3922db8
{"status":"using already created layer sha256:7c23fb36d80141c4ab8cdbb61ee479
{"status":"using already created layer sha256:2e0493f67d0c8c9c68a8aeacdf6a38
{"status":"using already created layer sha256:2759286baa875dc22de5394b4a9257
{"status":"writing layer sha256:df30045fe90f0d750db82a058109cecd6d4de9c90a3c
{"status":"writing layer sha256:f18a68eb09bf925bb1b669490407c1b1251c5db98dc4
{"status":"writing manifest"}
{"status":"success"}
```

Quantize a model

Quantize a non-quantized model.

Request

```
curl http://localhost:11434/api/create -d '{
  "model": "llama3.2:quantized",
  "from": "llama3.2:3b-instruct-fp16",
  "quantize": "q4_K_M"
}'
```

Response

A stream of JSON objects is returned:

```
{"status":"quantizing F16 model to Q4_K_M","digest":"0","total":6433687776,'
{"status":"quantizing F16 model to Q4_K_M","digest":"0","total":6433687776,'
{"status":"verifying conversion"}
{"status":"creating new layer sha256:fb7f4f211b89c6c4928ff4ddb73db9f9c0cfca3
{"status":"using existing layer sha256:966de95ca8a62200913e3f8bfbf84c849453
{"status":"using existing layer sha256:fcc5a6bec9daf9b561a68827b67ab6088e1db
{"status":"using existing layer sha256:a70ff7e570d97baaf4e62ac6e6ad9975e04ca
{"status":"using existing layer sha256:56bb8bd477a519ffa694fc449c2413c6f0e1c
{"status":"writing manifest"}
{"status":"success"}
```

Create a model from GGUF

Create a model from a GGUF file. The `files` parameter should be filled out with the file name and SHA256 digest of the GGUF file you wish to use. Use /api/blobs/:digest to push the GGUF file to the server before calling this API.

Request

```
curl http://localhost:11434/api/create -d '{
  "model": "my-gguf-model",
  "files": {
    "test.gguf": "sha256:432f310a77f4650a88d0fd59ecdd7cebed8d684bafea53cbff0
  }
}'
```

Response

A stream of JSON objects is returned:

```
{"status":"parsing GGUF"}
{"status":"using existing layer sha256:432f310a77f4650a88d0fd59ecdd7cebed8d0
{"status":"writing manifest"}
{"status":"success"}
```

Create a model from a Safetensors directory

The `files` parameter should include a dictionary of files for the safetensors model which includes the file names and SHA256 digest of each file. Use /api/blobs/:digest to first push each of the files to the server before calling this API. Files will remain in the cache until the Ollama server is restarted.

Request

```
curl http://localhost:11434/api/create -d '{
  "model": "fred",
  "files": {
    "config.json": "sha256:dd3443e529fb2290423a0c65c2d633e67b419d273f170259e
    "generation_config.json": "sha256:88effbb63300dbbc7390143fbbdd9d9fa50587
    "special_tokens_map.json": "sha256:b7455f0e8f00539108837bfa586c4fbf424e3
    "tokenizer.json": "sha256:bbc1904d35169c542dffbe1f7589a5994ec7426d9e5b60
    "tokenizer_config.json": "sha256:24e8a6dc2547164b7002e3125f10b415105644f
    "model.safetensors": "sha256:1ff795ff6a07e6a68085d206fb84417da2f083f6839
  }
}'
```

Response

A stream of JSON objects is returned:

```
{"status":"converting model"}
{"status":"creating new layer sha256:05ca5b813af4a53d2c2922933936e398958855c
{"status":"using autodetected template llama3-instruct"}
{"status":"using existing layer sha256:56bb8bd477a519ffa694fc449c2413c6f0e1c
{"status":"writing manifest"}
{"status":"success"}
```

Check if a Blob Exists

```
HEAD /api/blobs/:digest
```

Ensures that the file blob (Binary Large Object) used with create a model exists on the server. This checks your Ollama server and not ollama.com.

Query Parameters

- `digest`: the SHA256 digest of the blob

Examples

Request

```
curl -I http://localhost:11434/api/blobs/sha256:29fdb92e57cf0827ded04ae6461b
```

Response

Return 200 OK if the blob exists, 404 Not Found if it does not.

Push a Blob

```
POST /api/blobs/:digest
```

Push a file to the Ollama server to create a "blob" (Binary Large Object).

Query Parameters

- `digest`: the expected SHA256 digest of the file

Examples

Request

```
curl -T model.gguf -X POST http://localhost:11434/api/blobs/sha256:29fdb92e5
```

Response

Return 201 Created if the blob was successfully created, 400 Bad Request if the digest used is not expected.

List Local Models

```
GET /api/tags
```

List models that are available locally.

Examples

Request

```
curl http://localhost:11434/api/tags
```

Response

A single JSON object will be returned.


```
{
  "models": [
    {
      "name": "deepseek-r1:latest",
      "model": "deepseek-r1:latest",
      "modified_at": "2025-05-10T08:06:48.639712648-07:00",
      "size": 4683075271,
      "digest": "0a8c266910232fd3291e71e5ba1e058cc5af9d411192cf88b6d30e92b6e
      "details": {
        "parent_model": "",
        "format": "gguf",
        "family": "qwen2",
        "families": ["qwen2"],
        "parameter_size": "7.6B",
        "quantization_level": "Q4_K_M"
      }
    },
    {
      "name": "llama3.2:latest",
      "model": "llama3.2:latest",
      "modified_at": "2025-05-04T17:37:44.706015396-07:00",
      "size": 2019393189,
      "digest": "a80c4f17acd55265feec403c7aef86be0c25983ab279d83f3bcd3abbcb5
      "details": {
        "parent_model": "",
        "format": "gguf",
        "family": "llama",
        "families": ["llama"],
        "parameter_size": "3.2B",
        "quantization_level": "Q4_K_M"
      }
    }
  ]
}
```

Show Model Information

```
POST /api/show
```

Show information about a model including details, modelfile, template, parameters, license, system prompt.

Parameters

- `model`: name of the model to show
- `verbose`: (optional) if set to `true`, returns full data for verbose response fields

Examples

Request

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

Response

```
{
  modelfile: '# Modelfile generated by "ollama show"\n# To build a new Model
  parameters: 'num_keep                          24\nstop
  template: "{{ if .System }}<|start_header_id|>system<|end_header_id|>\n\n{
  details: {
    parent_model: "",
    format: "gguf",
    family: "llama",
    families: ["llama"],
    parameter_size: "8.0B",
    quantization_level: "Q4_0",
  },
  model_info: {
    "general.architecture": "llama",
    "general.file_type": 2,
    "general.parameter_count": 8030261248,
    "general.quantization_version": 2,
    "llama.attention.head_count": 32,
    "llama.attention.head_count_kv": 8,
    "llama.attention.layer_norm_rms_epsilon": 0.00001,
    "llama.block_count": 32,
```

```
    "llama.context_length": 8192,
    "llama.embedding_length": 4096,
    "llama.feed_forward_length": 14336,
    "llama.rope.dimension_count": 128,
    "llama.rope.freq_base": 500000,
    "llama.vocab_size": 128256,
    "tokenizer.ggml.bos_token_id": 128000,
    "tokenizer.ggml.eos_token_id": 128009,
    "tokenizer.ggml.merges": [], // populates if `verbose=true`
    "tokenizer.ggml.model": "gpt2",
    "tokenizer.ggml.pre": "llama-bpe",
    "tokenizer.ggml.token_type": [], // populates if `verbose=true`
    "tokenizer.ggml.tokens": [], // populates if `verbose=true`
  },
  capabilities: ["completion", "vision"],
}
```

Copy a Model

```
POST /api/copy
```

Copy a model. Creates a model with another name from an existing model.

Examples

Request

```
curl http://localhost:11434/api/copy -d '{
  "source": "llama3.2",
  "destination": "llama3-backup"
}'
```

Response

Returns a 200 OK if successful, or a 404 Not Found if the source model doesn't exist.

Delete a Model

```
DELETE /api/delete
```

Delete a model and its data.

Parameters

- `model`: model name to delete

Examples

Request

```
curl -X DELETE http://localhost:11434/api/delete -d '{
  "model": "llama3:13b"
}'
```

Response

Returns a 200 OK if successful, 404 Not Found if the model to be deleted doesn't exist.

Pull a Model

```
POST /api/pull
```

Download a model from the ollama library. Cancelled pulls are resumed from where they left off, and multiple calls will share the same download progress.

Parameters

- `model`: name of the model to pull
- `insecure`: (optional) allow insecure connections to the library. Only use this if you are pulling from your own library during development.
- `stream`: (optional) if `false` the response will be returned as a single response object, rather than a stream of objects

Examples

Request

```
curl http://localhost:11434/api/pull -d '{
  "model": "llama3.2"
```

```
}'
```

Response

If `stream` is not specified, or set to `true`, a stream of JSON objects is returned:

The first object is the manifest:

```
{
  "status": "pulling manifest"
}
```

Then there is a series of downloading responses. Until any of the download is completed, the `completed` key may not be included. The number of files to be downloaded depends on the number of layers specified in the manifest.

```
{
  "status": "pulling digestname",
  "digest": "digestname",
  "total": 2142590208,
  "completed": 241970
}
```

After all the files are downloaded, the final responses are:

```
{
    "status": "verifying sha256 digest"
}
{
    "status": "writing manifest"
}
{
    "status": "removing any unused layers"
}
{
    "status": "success"
}
```

if `stream` is set to false, then the response is a single JSON object:

```
{
  "status": "success"
```

```
}
```

Push a Model

```
POST /api/push
```

Upload a model to a model library. Requires registering for ollama.ai and adding a public key first.

Parameters

- `model`: name of the model to push in the form of `<namespace>/<model>:<tag>`
- `insecure`: (optional) allow insecure connections to the library. Only use this if you are pushing to your library during development.
- `stream`: (optional) if `false` the response will be returned as a single response object, rather than a stream of objects

Examples

Request

```
curl http://localhost:11434/api/push -d '{
  "model": "mattw/pygmalion:latest"
}'
```

Response

If `stream` is not specified, or set to `true`, a stream of JSON objects is returned:

```
{ "status": "retrieving manifest" }
```

and then:

```
{
  "status": "starting upload",
  "digest": "sha256:bc07c81de745696fdf5afca05e065818a8149fb0c77266fb584d9b2c
  "total": 1928429856
}
```

Then there is a series of uploading responses:

```
{
  "status": "starting upload",
  "digest": "sha256:bc07c81de745696fdf5afca05e065818a8149fb0c77266fb584d9b2c
  "total": 1928429856
}
```

Finally, when the upload is complete:

```
{"status":"pushing manifest"}
{"status":"success"}
```

If `stream` is set to `false`, then the response is a single JSON object:

```
{ "status": "success" }
```

Generate Embeddings

```
POST /api/embed
```

Generate embeddings from a model

Parameters

- `model`: name of model to generate embeddings from
- `input`: text or list of text to generate embeddings for

Advanced parameters:

- `truncate`: truncates the end of each input to fit within context length. Returns error if `false` and context length is exceeded. Defaults to `true`
- `options`: additional model parameters listed in the documentation for the Modelfile such as `temperature`
- `keep_alive`: controls how long the model will stay loaded into memory following the request (default: `5m`)
- `dimensions`: number of dimensions for the embedding

Examples

Request

```
curl http://localhost:11434/api/embed -d '{
  "model": "all-minilm",
  "input": "Why is the sky blue?"
}'
```

Response

```
{
  "model": "all-minilm",
  "embeddings": [
    [
      0.010071029, -0.0017594862, 0.05007221, 0.04692972, 0.054916814,
      0.008599704, 0.105441414, -0.025878139, 0.12958129, 0.031952348
    ]
  ],
  "total_duration": 14143917,
  "load_duration": 1019500,
  "prompt_eval_count": 8
}
```

Request (Multiple input)

```
curl http://localhost:11434/api/embed -d '{
  "model": "all-minilm",
  "input": ["Why is the sky blue?", "Why is the grass green?"]
}'
```

Response

```
{
  "model": "all-minilm",
  "embeddings": [
    [
      0.010071029, -0.0017594862, 0.05007221, 0.04692972, 0.054916814,
      0.008599704, 0.105441414, -0.025878139, 0.12958129, 0.031952348
    ],
    [
      -0.0098027075, 0.06042469, 0.025257962, -0.006364387, 0.07272725,
      0.017194884, 0.09032035, -0.051705178, 0.09951512, 0.09072481
```

```
    ]
  ]
}
```

List Running Models

```
GET /api/ps
```

List models that are currently loaded into memory.

Examples

Request

```
curl http://localhost:11434/api/ps
```

Response

A single JSON object will be returned.

```
{
  "models": [
    {
      "name": "mistral:latest",
      "model": "mistral:latest",
      "size": 5137025024,
      "digest": "2ae6f6dd7a3dd734790bbbf58b8909a606e0e7e97e94b7604e0aa7ae44
      "details": {
        "parent_model": "",
        "format": "gguf",
        "family": "llama",
        "families": ["llama"],
        "parameter_size": "7.2B",
        "quantization_level": "Q4_0"
      },
      "expires_at": "2024-06-04T14:38:31.83753-07:00",
      "size_vram": 5137025024
    }
  ]
}
```

Generate Embedding

> Note: this endpoint has been superseded by `/api/embed`

```
POST /api/embeddings
```

Generate embeddings from a model

Parameters

- `model`: name of model to generate embeddings from
- `prompt`: text to generate embeddings for

Advanced parameters:

- `options`: additional model parameters listed in the documentation for the Modelfile such as `temperature`
- `keep_alive`: controls how long the model will stay loaded into memory following the request (default: `5m`)

Examples

Request

```
curl http://localhost:11434/api/embeddings -d '{
  "model": "all-minilm",
  "prompt": "Here is an article about llamas..."
}'
```

Response

```
{
  "embedding": [
    0.5670403838157654, 0.009260174818336964, 0.23178744316101074,
    -0.2916173040866852, -0.8924556970596313, 0.8785552978515625,
    -0.34576427936553955, 0.5742510557174683, -0.04222835972905159,
    -0.137906014919281
  ]
}
```

Version

```
GET /api/version
```

Retrieve the Ollama version

Examples

Request

```
curl http://localhost:11434/api/version
```

Response

```
{
  "version": "0.5.1"
}
```