API

Endpoints

- Generate a completion
- Generate a chat completion
- Create a Model
- List Local Models
- Show Model Information
- Copy a Model
- Delete a Model
- Pull a Model
- Push a Model
- Generate Embeddings

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-q4_1` and `llama2:70b`. 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 and can optional return non-streamed responses.

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
- `images`: (optional) a list of base64-encoded images (for multimodal models such as `llava`)

Advanced parameters (optional):

- `format`: the format to return a response in. Currently the only accepted value is `json`
- `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`)
- `context`: the context parameter returned from a previous request to `/generate`, this can be used to keep a short conversational memory
- `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`)

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.

> Note: 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": "llama2",
  "prompt": "Why is the sky blue?"
}'
```

Response

A stream of JSON objects is returned:

```
{
  "model": "llama2",
  "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 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` .

```
{
  "model": "llama2",
  "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": "llama2",
  "prompt": "Why is the sky blue?",
  "stream": false
}'
```

Response

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

```
{
  "model": "llama2",
  "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 (JSON mode)

> 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": "llama2",
  "prompt": "What color is the sky at different times of the day? Respond usin
  "format": "json",
  "stream": false
}'
```

Response

```
{
  "model": "llama2",
  "created_at": "2023-11-09T21:07:55.186497Z",
  "response": "{\n\"morning\": {\n\"color\": \"blue\"\n},\n\"noon\": {\n\"colo
  "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+VAAAACXBIWXMAAAsTAAAL
}'
```

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 `temperature` to 0 and `seed` to a number:

Request

```
curl http://localhost:11434/api/generate -d '{
  "model": "mistral",
  "prompt": "[INST] why is the sky blue? [/INST]",
  "options": {
    "seed": 101,
    "temperature": 0
  }
}'
```

Response

```
{
  "model": "mistral",
  "created_at": "2023-11-03T15:36:02.583064Z",
  "response": " The sky appears blue because of a phenomenon called Rayleigh s
  "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": "llama2",
  "prompt": "Why is the sky blue?",
  "stream": false,
  "options": {
    "num_keep": 5,
    "seed": 42,
    "num_predict": 100,
    "top_k": 20,
    "top_p": 0.9,
    "tfs_z": 0.5,
    "typical_p": 0.7,
    "repeat_last_n": 33,
    "temperature": 0.8,
    "repeat_penalty": 1.2,
    "presence_penalty": 1.5,
    "frequency_penalty": 1.0,
    "mirostat": 1,
    "mirostat_tau": 0.8,
    "mirostat_eta": 0.6,
    "penalize_newline": true,
    "stop": ["\n", "user:"],
    "numa": false,
    "num_ctx": 1024,
    "num_batch": 2,
    "num_gqa": 1,
    "num_gpu": 1,
    "main_gpu": 0,
    "low_vram": false,
    "f16_kv": true,
    "vocab_only": false,
    "use_mmap": true,
    "use_mlock": false,
    "embedding_only": false,
    "rope_frequency_base": 1.1,
    "rope_frequency_scale": 0.8,
    "num_thread": 8
  }
}'
```

Response

```
{
  "model": "llama2",
  "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": "llama2"
}'
```

Response

A single JSON object is returned:

```
{
  "model": "llama2",
  "created_at": "2023-12-18T19:52:07.071755Z",
  "response": "",
  "done": true
}
```

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

The `message` object has the following fields:

- `role`: the role of the message, either `system`, `user` or `assistant`
- `content`: the content of the message
- `images` (optional): a list of images to include in the message (for multimodal models such as `llava`)

Advanced parameters (optional):

- `format`: the format to return a response in. Currently the only accepted value is `json`
- `options`: additional model parameters listed in the documentation for the Modelfile such as `temperature`
- `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
- `keep_alive`: controls how long the model will stay loaded into memory following the request (default: `5m`)

Examples

Chat Request (Streaming)

Request

Send a chat message with a streaming response.

```
curl http://localhost:11434/api/chat -d '{
  "model": "llama2",
  "messages": [
    {
      "role": "user",
      "content": "why is the sky blue?"
    }
  ]
}'
```

Response

A stream of JSON objects is returned:

```
{
  "model": "llama2",
  "created_at": "2023-08-04T08:52:19.385406455-07:00",
  "message": {
    "role": "assistant",
    "content": "The",
    "images": null
  },
  "done": false
}
```

Final response:

```
{
  "model": "llama2",
  "created_at": "2023-08-04T19:22:45.499127Z",
  "done": true,
  "total_duration": 4883583458,
  "load_duration": 1334875,
  "prompt_eval_count": 26,
  "prompt_eval_duration": 342546000,
  "eval_count": 282,
  "eval_duration": 4535599000
}
```

Chat request (No streaming)

Request

```
curl http://localhost:11434/api/chat -d '{
  "model": "llama2",
  "messages": [
    {
      "role": "user",
      "content": "why is the sky blue?"
    }
  ],
  "stream": false
}'
```

Response

```
{
  "model": "registry.ollama.ai/library/llama2:latest",
  "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 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": "llama2",
  "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": "llama2",
  "created_at": "2023-08-04T08:52:19.385406455-07:00",
  "message": {
    "role": "assistant",
    "content": "The"
  },
  "done": false
}
```

Final response:

```
{
  "model": "llama2",
  "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 images)

Request

Send a chat message with a conversation history.

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

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 ex
    "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": "llama2",
  "messages": [
    {
      "role": "user",
      "content": "Hello!"
    }
  ],
  "options": {
    "seed": 101,
    "temperature": 0
  }
}'
```

Response

```
{
  "model": "registry.ollama.ai/library/llama2:latest",
  "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
}
```

Create a Model

```
POST /api/create
```

Create a model from a `Modelfile`. It is recommended to set `modelfile` to the content of the Modelfile rather than just set `path`. This is a requirement for remote create. Remote model creation must also create any file blobs, fields such as `FROM` and `ADAPTER`, explicitly with the server using Create a Blob and the value to the path indicated in the response.

Parameters

- `name`: name of the model to create
- `modelfile` (optional): contents of the Modelfile
- `stream`: (optional) if `false` the response will be returned as a single response object, rather than a stream of objects
- `path` (optional): path to the Modelfile

Examples

Create a new model

Create a new model from a `Modelfile`.

Request

```
curl http://localhost:11434/api/create -d '{
  "name": "mario",
  "modelfile": "FROM llama2\nSYSTEM You are mario from Super Mario Bros."
}'
```

Response

A stream of JSON objects. Notice that the final JSON object shows a `"status": "success"`.

```
{"status":"reading model metadata"}
{"status":"creating system layer"}
{"status":"using already created layer sha256:22f7f8ef5f4c791c1b03d7eb41439929
{"status":"using already created layer sha256:8c17c2ebb0ea011be9981cc3922db8ca
{"status":"using already created layer sha256:7c23fb36d80141c4ab8cdbb61ee47901
{"status":"using already created layer sha256:2e0493f67d0c8c9c68a8aeacdf6a38a2
{"status":"using already created layer sha256:2759286baa875dc22de5394b4a925701
{"status":"writing layer sha256:df30045fe90f0d750db82a058109cecd6d4de9c90a3d75
{"status":"writing layer sha256:f18a68eb09bf925bb1b669490407c1b1251c5db98dc4d3
{"status":"writing manifest"}
{"status":"success"}
```

Check if a Blob Exists

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

Ensures that the file blob used for a FROM or ADAPTER field exists on the server. This is checking your Ollama server and not Ollama.ai.

Query Parameters

- `digest` : the SHA256 digest of the blob

Examples

Request

```
curl -I http://localhost:11434/api/blobs/sha256:29fdb92e57cf0827ded04ae6461b59
```

Response

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

Create a Blob

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

Create a blob from a file on the server. Returns the server file path.

Query Parameters

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

Examples

Request

```
curl -T model.bin -X POST http://localhost:11434/api/blobs/sha256:29fdb92e57cf
```

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": "codellama:13b",
      "modified_at": "2023-11-04T14:56:49.277302595-07:00",
      "size": 7365960935,
      "digest": "9f438cb9cd581fc025612d27f7c1a6669ff83a8bb0ed86c94fcf4c5440555
      "details": {
        "format": "gguf",
        "family": "llama",
        "families": null,
        "parameter_size": "13B",
        "quantization_level": "Q4_0"
      }
    },
    {
      "name": "llama2:latest",
      "modified_at": "2023-12-07T09:32:18.757212583-08:00",
      "size": 3825819519,
      "digest": "fe938a131f40e6f6d40083c9f0f430a515233eb2edaa6d72eb85c50d64f23
      "details": {
        "format": "gguf",
        "family": "llama",
        "families": null,
        "parameter_size": "7B",
        "quantization_level": "Q4_0"
      }
    }
  ]
}
```

Show Model Information

```
POST /api/show
```

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

Parameters

- `name` : name of the model to show

Examples

Request

```
curl http://localhost:11434/api/show -d '{
  "name": "llama2"
}'
```

Response

```
{
  "modelfile": "# Modelfile generated by \"ollama show\"\n# To build a new Mod
  "parameters": "num_ctx                        4096\nstop
  "template": "{{ .System }}\nUSER: {{ .Prompt }}\nASSSISTANT: ",
  "details": {
    "format": "gguf",
    "family": "llama",
    "families": ["llama", "clip"],
    "parameter_size": "7B",
    "quantization_level": "Q4_0"
  }
}
```

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": "llama2",
  "destination": "llama2-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

- `name` : model name to delete

Examples

Request

```
curl -X DELETE http://localhost:11434/api/delete -d '{
  "name": "llama2:13b"
}'
```

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

- `name` : 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 '{
  "name": "llama2"
}'
```

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": "downloading 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

- `name` : 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 '{
  "name": "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:bc07c81de745696fdf5afca05e065818a8149fb0c77266fb584d9b2cba
  "total": 1928429856
}
```

Then there is a series of uploading responses:

```
{
  "status": "starting upload",
  "digest": "sha256:bc07c81de745696fdf5afca05e065818a8149fb0c77266fb584d9b2cba
  "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/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": "llama2",
  "prompt": "Here is an article about llamas..."
}'
```

Response

```
{
  "embedding": [
    0.5670403838157654, 0.009260174818336964, 0.23178744316101074, -0.29161730
    0.8785552978515625, -0.34576427936553955, 0.5742510557174683, -0.042228359
  ]
}
```