[**https://github.com/ollama/ollama/blob/main/docs/api.md**](https://github.com/ollama/ollama/blob/main/docs/api.md)

**Date: 20/5/2024, Ollama version 0.1.38**

**OLLAMA**

**API**

**Endpoints**

* [Generate a completion](https://github.com/ollama/ollama/blob/main/docs/api.md#generate-a-completion)
* [Generate a chat completion](https://github.com/ollama/ollama/blob/main/docs/api.md#generate-a-chat-completion)
* [Create a Model](https://github.com/ollama/ollama/blob/main/docs/api.md#create-a-model)
* [List Local Models](https://github.com/ollama/ollama/blob/main/docs/api.md#list-local-models)
* [Show Model Information](https://github.com/ollama/ollama/blob/main/docs/api.md#show-model-information)
* [Copy a Model](https://github.com/ollama/ollama/blob/main/docs/api.md#copy-a-model)
* [Delete a Model](https://github.com/ollama/ollama/blob/main/docs/api.md#delete-a-model)
* [Pull a Model](https://github.com/ollama/ollama/blob/main/docs/api.md#pull-a-model)
* [Push a Model](https://github.com/ollama/ollama/blob/main/docs/api.md#push-a-model)
* [Generate Embeddings](https://github.com/ollama/ollama/blob/main/docs/api.md#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 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 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](https://github.com/ollama/ollama/blob/main/docs/api.md#model-names)
* 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](https://github.com/ollama/ollama/blob/main/docs/modelfile.md#valid-parameters-and-values) 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](https://github.com/ollama/ollama/blob/main/docs/api.md#request-json-mode) 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": "llama3",

"prompt": "Why is the sky blue?"

}'

**Response**

A stream of JSON objects is returned:

{

"model": "llama3",

"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",

"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",

"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",

"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": "llama3",

"prompt": "What color is the sky at different times of the day? Respond using JSON",

"format": "json",

"stream": false

}'

**Response**

{

"model": "llama3",

"created\_at": "2023-11-09T21:07:55.186497Z",

"response": "{\n\"morning\": {\n\"color\": \"blue\"\n},\n\"noon\": {\n\"color\": \"blue-gray\"\n},\n\"afternoon\": {\n\"color\": \"warm gray\"\n},\n\"evening\": {\n\"color\": \"orange\"\n}\n}\n",

"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": [""]

}'

**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": "Why is the sky blue?",

"options": {

"seed": 123,

"temperature": 0

}

}'

**Response**

{

"model": "mistral",

"created\_at": "2023-11-03T15:36:02.583064Z",

"response": " The sky appears blue because of a phenomenon called Rayleigh scattering.",

"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",

"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\_gpu": 1,

"main\_gpu": 0,

"low\_vram": false,

"f16\_kv": true,

"vocab\_only": false,

"use\_mmap": true,

"use\_mlock": false,

"num\_thread": 8

}

}'

**Response**

{

"model": "llama3",

"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"

}'

**Response**

A single JSON object is returned:

{

"model": "llama3",

"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](https://github.com/ollama/ollama/blob/main/docs/api.md#model-names)
* 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](https://github.com/ollama/ollama/blob/main/docs/modelfile.md#valid-parameters-and-values) 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)

**Examples**

**Chat Request (Streaming)**

**Request**

Send a chat message with a streaming response.

curl http://localhost:11434/api/chat -d '{

"model": "llama3",

"messages": [

{

"role": "user",

"content": "why is the sky blue?"

}

]

}'

**Response**

A stream of JSON objects is returned:

{

"model": "llama3",

"created\_at": "2023-08-04T08:52:19.385406455-07:00",

"message": {

"role": "assistant",

"content": "The",

"images": null

},

"done": false

}

Final response:

{

"model": "llama3",

"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": "llama3",

"messages": [

{

"role": "user",

"content": "why is the sky blue?"

}

],

"stream": false

}'

**Response**

{

"model": "registry.ollama.ai/library/llama3: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": "llama3",

"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",

"created\_at": "2023-08-04T08:52:19.385406455-07:00",

"message": {

"role": "assistant",

"content": "The"

},

"done": false

}

Final response:

{

"model": "llama3",

"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": [""]

}

]

}'

**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 expression. It's wearing a heart on its shirt and is waving in the air. This scene appears to be part of a drawing or sketching project.",

"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",

"messages": [

{

"role": "user",

"content": "Hello!"

}

],

"options": {

"seed": 101,

"temperature": 0

}

}'

**Response**

{

"model": "registry.ollama.ai/library/llama3: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](https://github.com/ollama/ollama/blob/main/docs/modelfile.md). 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](https://github.com/ollama/ollama/blob/main/docs/api.md#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 llama3\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 sha256:22f7f8ef5f4c791c1b03d7eb414399294764d7cc82c7e94aa81a1feb80a983a2"}

{"status":"using already created layer sha256:8c17c2ebb0ea011be9981cc3922db8ca8fa61e828c5d3f44cb6ae342bf80460b"}

{"status":"using already created layer sha256:7c23fb36d80141c4ab8cdbb61ee4790102ebd2bf7aeff414453177d4f2110e5d"}

{"status":"using already created layer sha256:2e0493f67d0c8c9c68a8aeacdf6a38a2151cb3c4c1d42accf296e19810527988"}

{"status":"using already created layer sha256:2759286baa875dc22de5394b4a925701b1896a7e3f8e53275c36f75a877a82c9"}

{"status":"writing layer sha256:df30045fe90f0d750db82a058109cecd6d4de9c90a3d75b19c09e5f64580bb42"}

{"status":"writing layer sha256:f18a68eb09bf925bb1b669490407c1b1251c5db98dc4d3d81f3088498ea55690"}

{"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/sha256:29fdb92e57cf0827ded04ae6461b5931d01fa595843f55d36f5b275a52087dd2

**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/sha256:29fdb92e57cf0827ded04ae6461b5931d01fa595843f55d36f5b275a52087dd2

**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": "9f438cb9cd581fc025612d27f7c1a6669ff83a8bb0ed86c94fcf4c5440555697",

"details": {

"format": "gguf",

"family": "llama",

"families": null,

"parameter\_size": "13B",

"quantization\_level": "Q4\_0"

}

},

{

"name": "llama3:latest",

"modified\_at": "2023-12-07T09:32:18.757212583-08:00",

"size": 3825819519,

"digest": "fe938a131f40e6f6d40083c9f0f430a515233eb2edaa6d72eb85c50d64f2300e",

"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": "llama3"

}'

**Response**

{

"modelfile": "# Modelfile generated by \"ollama show\"\n# To build a new Modelfile based on this one, replace the FROM line with:\n# FROM llava:latest\n\nFROM /Users/matt/.ollama/models/blobs/sha256:200765e1283640ffbd013184bf496e261032fa75b99498a9613be4e94d63ad52\nTEMPLATE \"\"\"{{ .System }}\nUSER: {{ .Prompt }}\nASSISTANT: \"\"\"\nPARAMETER num\_ctx 4096\nPARAMETER stop \"\u003c/s\u003e\"\nPARAMETER stop \"USER:\"\nPARAMETER stop \"ASSISTANT:\"",

"parameters": "num\_ctx 4096\nstop \u003c/s\u003e\nstop USER:\nstop ASSISTANT:",

"template": "{{ .System }}\nUSER: {{ .Prompt }}\nASSISTANT: ",

"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": "llama3",

"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**

* name: model name to delete

**Examples**

**Request**

curl -X DELETE http://localhost:11434/api/delete -d '{

"name": "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**

* 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": "llama3"

}'

**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:bc07c81de745696fdf5afca05e065818a8149fb0c77266fb584d9b2cba3711ab",

"total": 1928429856

}

Then there is a series of uploading responses:

{

"status": "starting upload",

"digest": "sha256:bc07c81de745696fdf5afca05e065818a8149fb0c77266fb584d9b2cba3711ab",

"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](https://github.com/ollama/ollama/blob/main/docs/modelfile.md#valid-parameters-and-values) 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

]

}