# Modelfile Reference


- Table of Contents
- Format
- Examples
  - Basic Modelfile
- Instructions
  - FROM (Required)
    - Build from existing model
    - Build from a Safetensors model
    - Build from a GGUF model
  - PARAMETER
    - Valid Parameters and Values
  - TEMPLATE
    - Template Variables
  - SYSTEM
  - LICENSE
  - MESSAGE
    - Valid roles
    - Example conversation
  - CAPABILITY
  - REQUIRES
- Notes


A Modelfile is the blueprint to create and share customized models using Ollama.

# Table of Contents


- Format

- Examples
- Instructions
  - FROM (Required)
    - Build from existing model
    - Build from a Safetensors model
    - Build from a GGUF model
  - PARAMETER
    - Valid Parameters and Values
  - TEMPLATE
    - Template Variables
  - SYSTEM
  - LICENSE
  - MESSAGE
  - CAPABILITY
- Notes


# Format

The format of the `Modelfile`:

```
# comment
INSTRUCTION arguments
```

| Instruction | Description |
| --- | --- |
| `FROM` (required) | Defines the base model to use. |
| `PARAMETER` | Sets the parameters for how Ollama will run the model. |
| `TEMPLATE` | The full prompt template to be sent to the model. |
| `SYSTEM` | Specifies the system message that will be set in the template. |
| `LICENSE` | Specifies the legal license. |
| `MESSAGE` | Specify message history. |
| `REQUIRES` | Specify the minimum version of Ollama required by the model. |
| `CAPABILITY` | Declare an additional model capability. |

# Examples

## Basic `Modelfile`

An example of a `Modelfile` creating a mario blueprint:

```
FROM llama3.2
# sets the temperature to 1 [higher is more creative, lower is more coherent]
PARAMETER temperature 1
# sets the context window size to 4096, this controls how many tokens the LLM can use as context to generate the next token
PARAMETER num_ctx 4096

# sets a custom system message to specify the behavior of the chat assistant
SYSTEM You are Mario from super mario bros, acting as an assistant.
```

To use this:

1. Save it as a file (e.g. `Modelfile`)
2. `ollama create choose-a-model-name -f <location of the file e.g. ./Modelfile>`
3. `ollama run choose-a-model-name`
4. Start using the model!

To view the Modelfile of a given model, use the `ollama show --modelfile` command.

```
ollama show --modelfile llama3.2
```

```
# Modelfile generated by "ollama show"
# To build a new Modelfile based on this one, replace the FROM line with:
# FROM llama3.2:latest
FROM /Users/pdevine/.ollama/models/blobs/sha256-00e1317cbf74d901080d7100f57580ba8dd8de57203072dc6f668324ba545f29
TEMPLATE """{{ if .System }}<|start_header_id|>system<|end_header_id|>

{{ .System }}<|eot_id|>{{ end }}{{ if .Prompt }}<|start_header_id|>user<|end_header_id|>

{{ .Prompt }}<|eot_id|>{{ end }}<|start_header_id|>assistant<|end_header_id|>
```

```
{{ .Response }}<|eot_id|>"""
PARAMETER stop "<|start_header_id|>"
PARAMETER stop "<|end_header_id|>"
PARAMETER stop "<|eot_id|>"
PARAMETER stop "<|reserved_special_token"
```

# Instructions

# FROM (Required)

The `FROM` instruction defines the base model to use when creating a model.

```
FROM <model name>:<tag>
```

## Build from existing model

```
FROM llama3.2
```

Browse available models

## Build from a Safetensors model

```
FROM <model directory>
```

The model directory should contain the Safetensors weights for a supported architecture.

## Build from a GGUF model

```
FROM ./ollama-model.gguf
```

The GGUF file location should be specified as an absolute path or relative to the `Modelfile` location.

For a split GGUF model, keep the original split filenames and use a wildcard that matches every shard. Shards can also be listed with separate `FROM` instructions.

```
FROM ./model-*.gguf
```

# PARAMETER

The `PARAMETER` instruction defines a parameter that can be set when the model is run.

```
PARAMETER <parameter> <parametervalue>
```

# Valid Parameters and Values

| Parameter | Description | Value Type | Example Usage |
|---|---|---|---|
| num_ctx | Sets the size of the context window used to generate the next token. (Default: 2048) | int | num_ctx 4096 |
| repeat_last_n | Sets how far back for the model to look back to prevent repetition. (Default: 64, 0 = disabled, -1 = num_ctx) | int | repeat_last_n 64 |
| repeat_penalty | Sets how strongly to penalize repetitions. A higher value (e.g., 1.5) will penalize repetitions more strongly, while a lower value (e.g., 0.9) will be more lenient. (Default: 1.0, disabled) | float | repeat_penalty 1.1 |
| temperature | The temperature of the model. Increasing the temperature will make the model answer more creatively. (Default: 0.8) | float | temperature 0.7 |
| seed | Sets the random number seed to use for generation. Setting this to a specific number will make the model generate the same text for the same prompt. (Default: 0) | int | seed 42 |
| stop | Sets the stop sequences to use. When this pattern is encountered the LLM will stop generating text and return. Multiple stop patterns may be set by specifying multiple separate `stop` parameters in a modelfile. | string | stop "AI assistant:" |
| num_predict | Maximum number of tokens to predict when generating text. (Default: -1, infinite generation) | int | num_predict 42 |

| Parameter | Description | Value Type | Example Usage |
| --- | --- | --- | --- |
| draft_num_predict | Maximum number of speculative draft tokens to predict per step when a draft model is available. Separate draft models default to 4; embedded MTP tensors require setting this parameter. Set to 0 to disable speculative drafting. | int | draft_num_predict 4 |
| top_k | Reduces the probability of generating nonsense. A higher value (e.g. 100) will give more diverse answers, while a lower value (e.g. 10) will be more conservative. (Default: 40) | int | top_k 40 |
| top_p | Works together with top-k. A higher value (e.g., 0.95) will lead to more diverse text, while a lower value (e.g., 0.5) will generate more focused and conservative text. (Default: 0.9) | float | top_p 0.9 |
| min_p | Alternative to the top*p, and aims to ensure a balance of quality and variety. The parameter _p* represents the minimum probability for a token to be considered, relative to the probability of the most likely token. For example, with *p*=0.05 and the most likely token having a probability of 0.9, logits with a value less than 0.045 are filtered out. (Default: 0.0) | float | min_p 0.05 |

# TEMPLATE

`TEMPLATE` of the full prompt template to be passed into the model. It may include (optionally) a system message, a user's message and the response from the model. Note: syntax may be model specific. Templates use Go [template syntax](#).

## Template Variables

| Variable | Description |
| --- | --- |
| `{{ .System }}` | The system message used to specify custom behavior. |
| `{{ .Prompt }}` | The user prompt message. |
| `{{ .Response }}` | The response from the model. When generating a response, text after this variable is omitted. |

```
TEMPLATE """{{ if .System }}<|im_start|>system
{{ .System }}<|im_end|>
{{ end }}{{ if .Prompt }}<|im_start|>user
{{ .Prompt }}<|im_end|>
{{ end }}<|im_start|>assistant
"""
```

# SYSTEM

The `SYSTEM` instruction specifies the system message to be used in the template, if applicable.

```
SYSTEM """<system message>"""
```

# LICENSE

The `LICENSE` instruction allows you to specify the legal license under which the model used with this Modelfile is shared or distributed.

```
LICENSE """
<license text>
"""
```

# MESSAGE

The `MESSAGE` instruction allows you to specify a message history for the model to use when responding. Use multiple iterations of the MESSAGE command to build up a conversation which will guide the model to answer in a similar way.

```
MESSAGE <role> <message>
```

## Valid roles

| Role | Description |
| --- | --- |
| system | Alternate way of providing the SYSTEM message for the model. |
| user | An example message of what the user could have asked. |
| assistant | An example message of how the model should respond. |

### Example conversation

```
MESSAGE user Is Toronto in Canada?
MESSAGE assistant yes
MESSAGE user Is Sacramento in Canada?
MESSAGE assistant no
MESSAGE user Is Ontario in Canada?
MESSAGE assistant yes
```

## CAPABILITY

Adds a capability to a GGUF or MLX model without removing inherited or inferred capabilities. Repeat the instruction to add more than one. Declare `decision` only for weights trained for System One's decision prompts:

```
CAPABILITY decision
```

## REQUIRES

The `REQUIRES` instruction allows you to specify the minimum version of Ollama required by the model.

```
REQUIRES <version>
```

The version should be a valid Ollama version (e.g. 0.14.0).

# Notes

- the **`Modelfile` is not case sensitive**. In the examples, uppercase instructions are used to make it easier to distinguish it from arguments.
- Instructions can be in any order. In the examples, the `FROM` instruction is first to keep it easily readable.

---

□□□ #2
□ □□□ □□□ 2026-10-31 13:18:31 UTC
□ □□□ □□□ 2026-10-31 13:18:32 UTC