Ollama Run LLMs locally

START HERE

`ollama pull <model>`
Download a model

`ollama run <model>`
Chat with a model

`ollama run <model> "<prompt>"`
One-shot answer

`ollama ls`
List local models

`ollama ps`
What is loaded now?

`ollama stop <model>`
Unload a model

`ollama rm <model>`
Delete a model

`ollama serve`
Start the server

Registry ollama.com → `pull` → **Local models** ollama ls → `run` → **Server** port 11434 → `HTTP API` → **Clients** CLI, curl, apps

CONCEPTS

The CLI is a client: it talks to a local server on port 11434. Desktop and systemd installs start it for you.

Models stay in memory for 5 minutes after the last request, then unload. Change it with OLLAMA_KEEP_ALIVE.

Model names are name:tag; without a tag you get latest. Pick a tag to choose a size or quantization.

RUN AND CHAT

`ollama run <model>`
Start an interactive chat

`ollama run <model> "<prompt>"`
Answer one prompt and exit

`cat <file> | ollama run <model> "<prompt>"`
Feed a file as context

`ollama run <model> --verbose`
Show timings and tokens/s

`ollama run <model> --format json`
Force JSON output

`ollama run <model> --think=false`
Turn thinking off or set its effort
Takes true/false, or high/medium/low on models that support it.

`ollama run <model> --hidethinking`
Hide the thinking output

`ollama run <model> --keepalive <dur>`
Keep the model loaded, e.g. 30m

`ollama run <embed-model> "<text>"`
Print an embedding as JSON

`"""`
Start and end multi-line input
Type it on its own at the >>> prompt.

`ollama launch <app>`
Start a coding app wired to Ollama

MANAGE MODELS

`ollama pull <model>`
Download or update a model

`ollama ls`
List downloaded models

`ollama ps`
List loaded models

MANAGE MODELS (cont.)

`ollama stop <model>`
Unload a running model

`ollama rm <model>`
Remove a model

`ollama show <model>`
Show details of a model

`ollama show <model> --modelfile`
Print its Modelfile

`ollama show <model> --parameters`
Print its parameters

`ollama show <model> --template`
Print its prompt template

`ollama show <model> --system`
Print its system message

`ollama cp <source> <new-name>`
Copy a model under a new name

`ollama create <name> -f <Modelfile>`
Build a model from a Modelfile

`ollama push <user>/<model>`
Upload a model to ollama.com
The name must include your ollama.com username.

`ollama signin`
Sign in to ollama.com

IN-CHAT COMMANDS

`/? / /help`
Show help for slash commands

`/bye`
Leave the chat

`/clear`
Clear the session context

`/set system <text>`
Set the system message

`/set parameter <name> <value>`
Change a parameter, e.g. temperature

`/set verbose / /set quiet`
Show or hide stats after replies

`/set think / /set nothink`
Turn thinking on or off

`/set format json / /set noformat`
Toggle JSON output

`/show info`
Show model details

`/show modelfile`
Show the Modelfile

`/show parameters`
Show the parameters

IN-CHAT COMMANDS (cont.)

`/save <name>`
Save this session as a model

`/load <name>`
Load a saved session or a model

`/? shortcuts`
Show keyboard shortcuts

MODELFILE

Custom model

```
FROM llama3.2
PARAMETER temperature 0.7
PARAMETER num_ctx 8192
SYSTEM """You are a concise assistant."""
```

`FROM <model or path>`
Base model, GGUF file or weights dir

`PARAMETER <name> <value>`
Set a runtime parameter

`SYSTEM """<text>"""`
Set the system message

`MESSAGE <role> <text>`
Add an example turn (user or assistant)

`TEMPLATE """<go template>"""`
Define the full prompt format
Uses Go template variables .System, .Prompt and .Response; the right syntax depends on the model.

`PARAMETER stop "<seq>"`
Stop generating at a string

`PARAMETER num_ctx <n>`
Context window in tokens

`PARAMETER seed <n>`
Make output reproducible

SERVER AND ENVIRONMENT

`ollama serve`
Run the server in the foreground

`OLLAMA_HOST=0.0.0.0:11434`
Listen on all interfaces

`OLLAMA_HOST=<host:port> ollama ls`
Point the CLI at another server

`OLLAMA_MODELS=<dir>`
Where models are stored

`OLLAMA_KEEP_ALIVE=<dur>`
How long idle models stay loaded
10m, 24h or seconds; a negative value keeps them loaded, 0 unloads right away.

`OLLAMA_CONTEXT_LENGTH=<n>`
Default context window in tokens

SERVER AND ENVIRONMENT (cont.)

`OLLAMA_NUM_PARALLEL=<n>`
Parallel requests per model
Memory use grows with this times the context length.

`OLLAMA_MAX_LOADED_MODELS=<n>`
Models kept loaded at once

`OLLAMA_ORIGINS=<origin>`
Allow extra web origins (CORS)

`systemctl edit ollama.service`
Set variables for the Linux service
Add Environment="OLLAMA_X=y" under [Service], then systemctl daemon-reload and systemctl restart ollama.

REST API

`http://localhost:11434/api`
Base URL of the local API

`POST /api/chat`
Chat with message history

`POST /api/generate`
Complete a single prompt

`GET /api/tags`
List local models

`http://localhost:11434/v1`
OpenAI-compatible base URL

Responses stream by default; send "stream": false for one JSON reply.

RECIPES

Ask over HTTP

```
curl http://localhost:11434/api/chat -d '{
  "model": "<model>",
  "messages": [{"role": "user", "content": "Hello"}],
  "stream": false
}' | jq -r .message.content
```

Expose on the network (systemd)

```
sudo systemctl edit ollama.service
# add under [Service]:
# Environment="OLLAMA_HOST=0.0.0.0:11434"
sudo systemctl daemon-reload
sudo systemctl restart ollama
```

Make a persona model

```
printf 'FROM llama3.2\nSYSTEM "Answer in one line."\n' > Modelfile
ollama create terse -f Modelfile
ollama run terse "Why is the sky blue?"
```

Summarise a file

```
cat notes.md | ollama run <model> "Summarise in 5 bullets"
ollama ps
ollama stop <model>
```