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

**Date: 20/5/2024 Ollama Version 0.1.38**

**FAQ**

**How can I upgrade Ollama?**

Ollama on macOS and Windows will automatically download updates. Click on the taskbar or menubar item and then click "Restart to update" to apply the update. Updates can also be installed by downloading the latest version [manually](https://ollama.com/download/).

On Linux, re-run the install script:

curl -fsSL https://ollama.com/install.sh | sh

**How can I view the logs?**

Review the [Troubleshooting](https://github.com/ollama/ollama/blob/main/docs/troubleshooting.md) docs for more about using logs.

**Is my GPU compatible with Ollama?**

Please refer to the [GPU docs](https://github.com/ollama/ollama/blob/main/docs/gpu.md).

**How can I specify the context window size?**

By default, Ollama uses a context window size of 2048 tokens.

To change this when using ollama run, use /set parameter:

/set parameter num\_ctx 4096

When using the API, specify the num\_ctx parameter:

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

"model": "llama3",

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

"options": {

"num\_ctx": 4096

}

}'

**How do I configure Ollama server?**

Ollama server can be configured with environment variables.

**Setting environment variables on Mac**

If Ollama is run as a macOS application, environment variables should be set using launchctl:

1. For each environment variable, call launchctl setenv.

launchctl setenv OLLAMA\_HOST "0.0.0.0"

1. Restart Ollama application.

**Setting environment variables on Linux**

If Ollama is run as a systemd service, environment variables should be set using systemctl:

1. Edit the systemd service by calling systemctl edit ollama.service. This will open an editor.
2. For each environment variable, add a line Environment under section [Service]:
3. [Service]

Environment="OLLAMA\_HOST=0.0.0.0"

1. Save and exit.
2. Reload systemd and restart Ollama:
3. systemctl daemon-reload

systemctl restart ollama

**Setting environment variables on Windows**

On Windows, Ollama inherits your user and system environment variables.

1. First Quit Ollama by clicking on it in the task bar.
2. Start the Settings (Windows 11) or Control Panel (Windows 10) application and search for *environment variables*.
3. Click on *Edit environment variables for your account*.
4. Edit or create a new variable for your user account for OLLAMA\_HOST, OLLAMA\_MODELS, etc.
5. Click OK/Apply to save.
6. Start the Ollama application from the Windows Start menu.

**How can I expose Ollama on my network?**

Ollama binds 127.0.0.1 port 11434 by default. Change the bind address with the OLLAMA\_HOST environment variable.

Refer to the section [above](https://github.com/ollama/ollama/blob/main/docs/faq.md#how-do-i-configure-ollama-server) for how to set environment variables on your platform.

**How can I use Ollama with a proxy server?**

Ollama runs an HTTP server and can be exposed using a proxy server such as Nginx. To do so, configure the proxy to forward requests and optionally set required headers (if not exposing Ollama on the network). For example, with Nginx:

server {

listen 80;

server\_name example.com; # Replace with your domain or IP

location / {

proxy\_pass http://localhost:11434;

proxy\_set\_header Host localhost:11434;

}

}

**How can I use Ollama with ngrok?**

Ollama can be accessed using a range of tools for tunneling tools. For example with Ngrok:

ngrok http 11434 --host-header="localhost:11434"

**How can I use Ollama with Cloudflare Tunnel?**

To use Ollama with Cloudflare Tunnel, use the --url and --http-host-header flags:

cloudflared tunnel --url http://localhost:11434 --http-host-header="localhost:11434"

**How can I allow additional web origins to access Ollama?**

Ollama allows cross-origin requests from 127.0.0.1 and 0.0.0.0 by default. Additional origins can be configured with OLLAMA\_ORIGINS.

Refer to the section [above](https://github.com/ollama/ollama/blob/main/docs/faq.md#how-do-i-configure-ollama-server) for how to set environment variables on your platform.

**Where are models stored?**

* macOS: ~/.ollama/models
* Linux: /usr/share/ollama/.ollama/models
* Windows: C:\Users\%username%\.ollama\models

**How do I set them to a different location?**

If a different directory needs to be used, set the environment variable OLLAMA\_MODELS to the chosen directory.

Refer to the section [above](https://github.com/ollama/ollama/blob/main/docs/faq.md#how-do-i-configure-ollama-server) for how to set environment variables on your platform.

**Does Ollama send my prompts and answers back to ollama.com?**

No. Ollama runs locally, and conversation data does not leave your machine.

**How can I use Ollama in Visual Studio Code?**

There is already a large collection of plugins available for VSCode as well as other editors that leverage Ollama. See the list of [extensions & plugins](https://github.com/ollama/ollama#extensions--plugins) at the bottom of the main repository readme.

**How do I use Ollama behind a proxy?**

Ollama is compatible with proxy servers if HTTP\_PROXY or HTTPS\_PROXY are configured. When using either variables, ensure it is set where ollama serve can access the values. When using HTTPS\_PROXY, ensure the proxy certificate is installed as a system certificate. Refer to the section above for how to use environment variables on your platform.

**How do I use Ollama behind a proxy in Docker?**

The Ollama Docker container image can be configured to use a proxy by passing -e HTTPS\_PROXY=https://proxy.example.com when starting the container.

Alternatively, the Docker daemon can be configured to use a proxy. Instructions are available for Docker Desktop on [macOS](https://docs.docker.com/desktop/settings/mac/#proxies), [Windows](https://docs.docker.com/desktop/settings/windows/#proxies), and [Linux](https://docs.docker.com/desktop/settings/linux/#proxies), and Docker [daemon with systemd](https://docs.docker.com/config/daemon/systemd/#httphttps-proxy).

Ensure the certificate is installed as a system certificate when using HTTPS. This may require a new Docker image when using a self-signed certificate.

FROM ollama/ollama

COPY my-ca.pem /usr/local/share/ca-certificates/my-ca.crt

RUN update-ca-certificates

Build and run this image:

docker build -t ollama-with-ca .

docker run -d -e HTTPS\_PROXY=https://my.proxy.example.com -p 11434:11434 ollama-with-ca

**How do I use Ollama with GPU acceleration in Docker?**

The Ollama Docker container can be configured with GPU acceleration in Linux or Windows (with WSL2). This requires the [nvidia-container-toolkit](https://github.com/NVIDIA/nvidia-container-toolkit). See [ollama/ollama](https://hub.docker.com/r/ollama/ollama) for more details.

GPU acceleration is not available for Docker Desktop in macOS due to the lack of GPU passthrough and emulation.

**Why is networking slow in WSL2 on Windows 10?**

This can impact both installing Ollama, as well as downloading models.

Open Control Panel > Networking and Internet > View network status and tasks and click on Change adapter settings on the left panel. Find the vEthernel (WSL) adapter, right click and select Properties. Click on Configure and open the Advanced tab. Search through each of the properties until you find Large Send Offload Version 2 (IPv4) and Large Send Offload Version 2 (IPv6). *Disable* both of these properties.

**How can I pre-load a model to get faster response times?**

If you are using the API you can preload a model by sending the Ollama server an empty request. This works with both the /api/generate and /api/chat API endpoints.

To preload the mistral model using the generate endpoint, use:

curl http://localhost:11434/api/generate -d '{"model": "mistral"}'

To use the chat completions endpoint, use:

curl http://localhost:11434/api/chat -d '{"model": "mistral"}'

**How do I keep a model loaded in memory or make it unload immediately?**

By default models are kept in memory for 5 minutes before being unloaded. This allows for quicker response times if you are making numerous requests to the LLM. You may, however, want to free up the memory before the 5 minutes have elapsed or keep the model loaded indefinitely. Use the keep\_alive parameter with either the /api/generate and /api/chat API endpoints to control how long the model is left in memory.

The keep\_alive parameter can be set to:

* a duration string (such as "10m" or "24h")
* a number in seconds (such as 3600)
* any negative number which will keep the model loaded in memory (e.g. -1 or "-1m")
* '0' which will unload the model immediately after generating a response

For example, to preload a model and leave it in memory use:

curl http://localhost:11434/api/generate -d '{"model": "llama3", "keep\_alive": -1}'

To unload the model and free up memory use:

curl http://localhost:11434/api/generate -d '{"model": "llama3", "keep\_alive": 0}'

Alternatively, you can change the amount of time all models are loaded into memory by setting the OLLAMA\_KEEP\_ALIVE environment variable when starting the Ollama server. The OLLAMA\_KEEP\_ALIVE variable uses the same parameter types as the keep\_alive parameter types mentioned above. Refer to section explaining [how to configure the Ollama server](https://github.com/ollama/ollama/blob/main/docs/faq.md#how-do-i-configure-ollama-server) to correctly set the environment variable.

If you wish to override the OLLAMA\_KEEP\_ALIVE setting, use the keep\_alive API parameter with the /api/generate or /api/chat API endpoints.

**How do I manage the maximum number of requests the server can queue**

If too many requests are sent to the server, it will respond with a 503 error indicating the server is overloaded. You can adjust how many requests may be queue by setting OLLAMA\_MAX\_QUEUE