FAQ

- How can I view the logs?
- Is my GPU compatible with Ollama?
- How can I specify the context window size?
- How can I tell if my model was loaded onto the GPU?
- How do I configure Ollama server?
  - Setting environment variables on Mac
  - Setting environment variables on Linux
  - Setting environment variables on Windows
- How do I use Ollama behind a proxy?
  - How do I use Ollama behind a proxy in Docker?
- Does Ollama send my prompts and answers back to ollama.com?
- How do I disable Ollama's cloud features? {#how-do-i-disable-ollama-cloud-features}
- How can I expose Ollama on my network?
- How can I use Ollama with a proxy server?
- How can I use Ollama with ngrok?
- How can I use Ollama with Cloudflare Tunnel?
- How can I allow additional web origins to access Ollama?
- Where are models stored?
  - How do I set them to a different location?
- How can I use Ollama in Visual Studio Code?
- How do I use Ollama with GPU acceleration in Docker?
- Why is networking slow in WSL2 on Windows 10?
- How can I preload a model into Ollama to get faster response times?
- How do I keep a model loaded in memory or make it unload immediately?
- How do I manage the maximum number of requests the Ollama server can queue?
- How does Ollama handle concurrent requests?
- How does Ollama load models on multiple GPUs?
- How can I enable Flash Attention?

- How can I set the quantization type for the K/V cache?
- Where can I find my Ollama Public Key?
  - How to Add the Key
  - Where the Ollama Public Key lives
- How can I stop Ollama from starting when I login to my computer?

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.

On Linux, re-run the install script:

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

How can I view the logs?

Review the Troubleshooting docs for more about using logs.

Is my GPU compatible with Ollama?

Please refer to the GPU docs.

How can I specify the context window size?

By default, Ollama uses a context window size of 4096 tokens.

This can be overridden with the `OLLAMA_CONTEXT_LENGTH` environment variable. For example, to set the default context window to 8K, use:

```
OLLAMA_CONTEXT_LENGTH=8192 ollama serve
```

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": "llama3.2",
  "prompt": "Why is the sky blue?",
  "options": {
    "num_ctx": 4096
  }
}'
```

How can I tell if my model was loaded onto the GPU?

Use the `ollama ps` command to see what models are currently loaded into memory.

```
ollama ps
```

Output:

```
NAME        ID            SIZE    PROCESSOR   UNTIL
llama3:70b  bcfb190ca3a7  42 GB   100% GPU    4 minutes from now
```

The `Processor` column will show which memory the model was loaded into:

- `100% GPU` means the model was loaded entirely into the GPU
- `100% CPU` means the model was loaded entirely in system memory
- `48%/52% CPU/GPU` means the model was loaded partially onto both the GPU and into system memory

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 "0.0.0.0:11434"
   ```

2. 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]`:

   ```
   [Service]
   Environment="OLLAMA_HOST=0.0.0.0:11434"
   ```

3. Save and exit.
4. Reload `systemd` and restart Ollama:

   ```
   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 do I use Ollama behind a proxy?

Ollama pulls models from the Internet and may require a proxy server to access the models. Use `HTTPS_PROXY` to redirect outbound requests through the 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.

Avoid setting `HTTP_PROXY`. Ollama does not use HTTP for model pulls, only HTTPS. Setting `HTTP_PROXY` may interrupt client connections to the server.

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, Windows, and Linux, and Docker daemon with systemd.

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
```

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

Ollama runs locally. We don't see your prompts or data when you run locally. When using cloud-hosted models, we process your prompts and responses to provide the service but do not store or log that content and never train on it. We collect basic account info and limited usage metadata to provide the service that does not include prompt or response content. We don't sell your data. You can delete your account anytime.

How do I disable Ollama's cloud features? {#how-do-i-disable-ollama-cloud-features}

Ollama can run in local only mode by disabling Ollama's cloud features. By turning off Ollama's cloud features, you will lose the ability to use Ollama's cloud models and web search.

Set `disable_ollama_cloud` in `~/.ollama/server.json`:

```
{
  "disable_ollama_cloud": true
}
```

You can also set the environment variable:

```
OLLAMA_NO_CLOUD=1
```

Restart Ollama after changing configuration. Once disabled, Ollama's logs will show `Ollama cloud disabled: true`.

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 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 tunneling apps. 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`.

For browser extensions, you'll need to explicitly allow the extension's origin pattern. Set `OLLAMA_ORIGINS` to include `chrome-extension://*`, `moz-extension://*`, and `safari-web-extension://*` if you wish to allow all browser extensions access, or specific extensions as needed:

```
# Allow all Chrome, Firefox, and Safari extensions
OLLAMA_ORIGINS=chrome-extension://*,moz-extension://*,safari-web-extension://* ollama serve
```

Refer to the section above 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.

On Linux using the standard installer, the `ollama` user needs read and write access to the specified directory. To assign the directory to the `ollama` user run `sudo chown -R ollama:ollama `.

Refer to the section above for how to set environment variables on your platform.

How can I use Ollama in Visual Studio Code?

Install the Ollama extension to use Ollama models in VS Code Chat. See the VS Code integration guide for setup and troubleshooting.

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. See 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 `vEthernet (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 preload a model into Ollama 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"}'
```

To preload a model using the CLI, use the command:

```
ollama run llama3.2 ""
```

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're making numerous requests to the LLM. If you want to immediately unload a model from memory, use the `ollama stop` command:

```
ollama stop llama3.2
```

If you're using the API, use the `keep_alive` parameter with the `/api/generate` and `/api/chat` endpoints to set the amount of time that a model stays 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": "llama3.2", "keep_alive": -1}'
```

To unload the model and free up memory use:

```
curl http://localhost:11434/api/generate -d '{"model": "llama3.2", "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 the section explaining how to configure the Ollama server to correctly set the environment variable.

The `keep_alive` API parameter with the `/api/generate` and `/api/chat` API endpoints will override the `OLLAMA_KEEP_ALIVE` setting.

How do I manage the maximum number of requests the Ollama 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 queued by setting `OLLAMA_MAX_QUEUE`.

How does Ollama handle concurrent requests?

Ollama supports two levels of concurrent processing. If your system has sufficient available memory (system memory when using CPU inference, or VRAM for GPU inference) then multiple models can be loaded at the same time. For a given model, if there is sufficient available memory when the model is loaded, it is configured to allow parallel request processing.

If there is insufficient available memory to load a new model request while one or more models are already loaded, all new requests will be queued until the new model can be loaded. As prior models become idle, one or more will be unloaded to make room for the new model. Queued requests will be processed in order. When using GPU inference new models must be able to completely fit in VRAM to allow concurrent model loads.

Parallel request processing for a given model results in increasing the context size by the number of parallel requests. For example, a 2K context with 4 parallel requests will result in an 8K context and additional memory allocation.

The following server settings may be used to adjust how Ollama handles concurrent requests on most platforms:

- `OLLAMA_MAX_LOADED_MODELS` - The maximum number of models that can be loaded concurrently provided they fit in available memory. The default is 3 * the number of GPUs or 3 for CPU inference.
- `OLLAMA_NUM_PARALLEL` - The maximum number of parallel requests each model will process at the same time, default 1. Required RAM will scale by `OLLAMA_NUM_PARALLEL` * `OLLAMA_CONTEXT_LENGTH`.
- `OLLAMA_MAX_QUEUE` - The maximum number of requests Ollama will queue when busy before rejecting additional requests. The default is 512

Note: Windows with Radeon GPUs currently default to 1 model maximum due to limitations in ROCm v5.7 for available VRAM reporting. Once ROCm v6.2 is available, Windows Radeon will follow the defaults above. You may enable concurrent model loads on Radeon on Windows, but ensure you don't load more models than will fit into your GPU's VRAM.

How does Ollama load models on multiple GPUs?

When loading a new model, Ollama evaluates the required VRAM for the model against what is currently available. If the model will entirely fit on any single GPU, Ollama will load the model on that GPU. This typically provides the best performance as it reduces the amount of data transferring across the PCI bus during inference. If the model does not fit entirely on one GPU, then it will be spread across all the available GPUs.

How can I enable Flash Attention?

Flash Attention is a feature of most modern models that can significantly reduce memory usage as the context size grows. Ollama uses Flash Attention automatically when the selected backend and devices support it. To force Flash Attention on, set `OLLAMA_FLASH_ATTENTION=1` when starting the Ollama server. To disable it, set `OLLAMA_FLASH_ATTENTION=0`.

How can I set the quantization type for the K/V cache?

The K/V context cache can be quantized to significantly reduce memory usage when Flash Attention is enabled.

To use quantized K/V cache with Ollama you can set the following environment variable:

- `OLLAMA_KV_CACHE_TYPE` - The quantization type for the K/V cache. Default is `f16`.

Currently this is a global option - meaning all models will run with the specified quantization type. The currently available K/V cache quantization types are:

- `f16` - high precision and memory usage (default).
- `q8_0` - 8-bit quantization, uses approximately 1/2 the memory of `f16` with a very small loss in precision, this usually has no noticeable impact on the model's quality (recommended if not using f16).
- `q4_0` - 4-bit quantization, uses approximately 1/4 the memory of `f16` with a small-medium loss in precision that may be more noticeable at higher context sizes.

How much the cache quantization impacts the model's response quality will depend on the model and the task. Models that have a high GQA count (e.g. Qwen2) may see a larger impact on precision from quantization than models with a low GQA count.

You may need to experiment with different quantization types to find the best balance between memory usage and quality.

Where can I find my Ollama Public Key?

Your **Ollama Public Key** is the public part of the key pair that lets your local Ollama instance talk to ollama.com.

You'll need it to:

- Push models to Ollama
- Pull private models from Ollama to your machine
- Run models hosted in Ollama's cloud

How to Add the Key

- **Sign-in via the Settings page** in the **Mac** and **Windows App**
- **Sign-in via CLI**

```
ollama signin
```

- **Manually copy & paste** the key on the **Ollama Keys** page:
  https://ollama.com/settings/keys

Where the Ollama Public Key lives

OS	Path to `id_ed25519.pub`
macOS	`~/.ollama/id_ed25519.pub`
Linux	`/usr/share/ollama/.ollama/id_ed25519.pub`
Windows	`C:\Users\<username>\.ollama\id_ed25519.pub`

Replace <username> with your actual Windows user name.

How can I stop Ollama from starting when I login to my computer?

Ollama for Windows and macOS register as a login item during installation. You can disable this if you prefer not to have Ollama automatically start. Ollama will respect this setting across upgrades, unless you uninstall the application.

Windows

- In `Task Manager` go to the `Startup apps` tab, search for `ollama` then click `Disable`

MacOS

- Open `Settings` and search for "Login Items", find the `Ollama` entry under `Allow in the Background`, then click the slider to disable.

□□□ #1
□ □□□ □□□ 2026-10-31 13:17:15 UTC
□ □□□ □□□ 2026-10-31 13:17:15 UTC