Claris

Running your own AI Server

07/11/25

Why run your own FileMaker AI Model Server?

The reasons for running your own AI server instead of using commercial AI providers are covered extensively in other places, but fundamentally it is a choice centered on security and privacy. By keeping AI services, models, and workflows within a trusted network, you remain in control over your data and intellectual property.

This document aims to help you make the right hardware and infrastructure decisions when it comes to deploying your own AI Model Server provided with FileMaker Server and help you get underway on your AI journey.

TL; DR

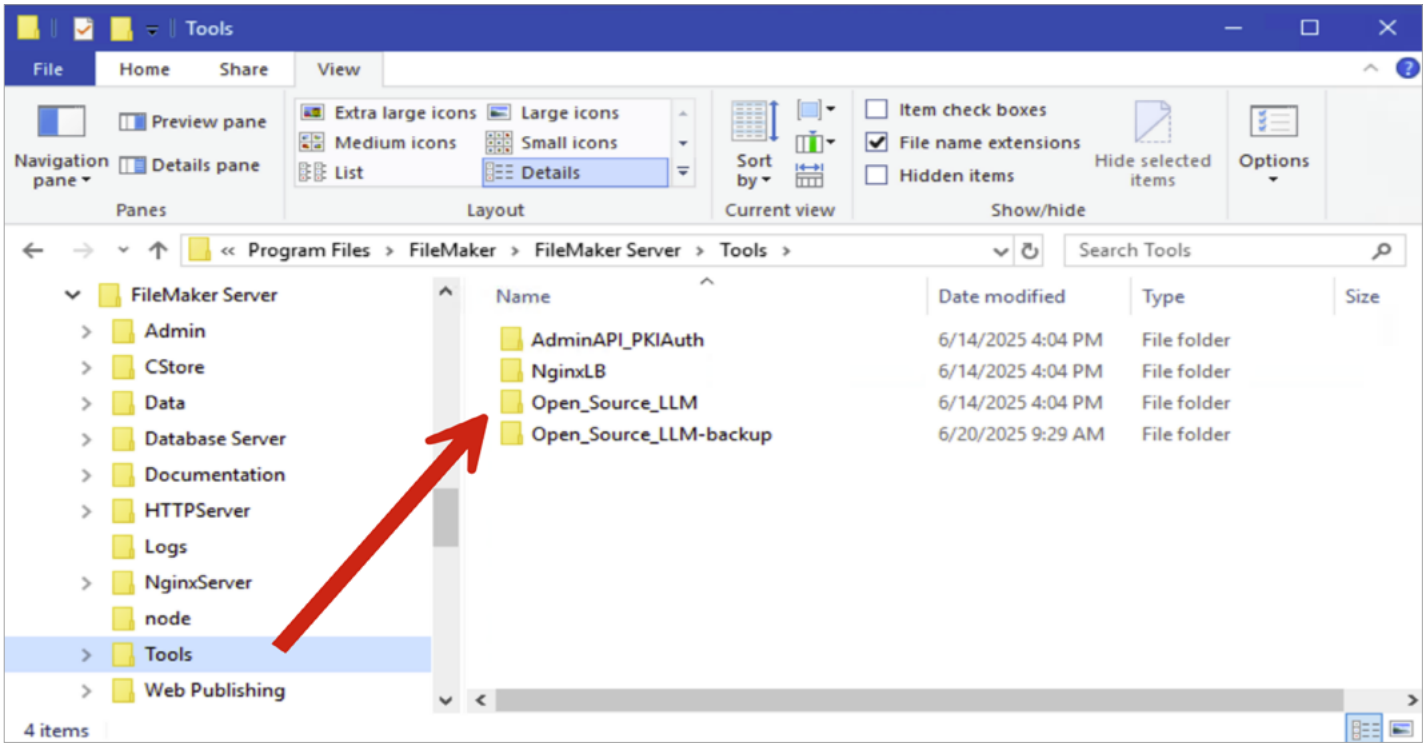
Here are the main takeaways from this white paper. The reasons and terminology used here are explained in detail in the document.

- **To prevent performance issues, the AI Model Server should run on a separate server, not your main FileMaker Server.**
  - Use a dedicated server with the right resources for an optimal experience. If needed, use the available feature flag to disable the AI Services page in Admin Console on your main FileMaker server
- **Text-embedding operations are generally less demanding than other inference or fine-tuning operations; size the machine for your AI Model Server accordingly.**
- **The AI Model Server needs a machine with GPUs.**
  - GGUF CPU-only models are not supported
  - On Windows and Ubuntu: use NVIDIA GPUs and enable CUDA
  - For macOS, use a Mac with Apple silicon (M-series chips) and pick an MLX model or convert a model to MLX
- **The size of an AI model (including LLMs) is directly tied to its quality and speed of response, but quantization on larger models can help reduce the size and maintain a workable level of performance.**
- **Prioritize having sufficient memory when choosing hardware resources.**
  - On a Mac with Apple silicon: 32 GB of RAM will get you going with the smaller models. 64-96 GB or more will give you access to bigger and more accurate resources
  - On Windows and Ubuntu computers: pay attention to the amount of VRAM on the NVIDIA hardware, more is better and a minimum of 24GB will work for small to medium models.
- **When selecting quantized models, as a rule of thumb, we recommend not going below 8-bit models for most AI workloads.**
- **On CUDA-enabled machines (Windows / Linux) where you want to use 8-bit quantization on-the-fly, make sure the machine has sufficient extra resources. On marginal hardware the effects might be noticeable.**
- **The AI Model Server can load one model of each type (text embedding, image embedding, and text generation) at a time.**
- **Avoid calling different models of the same type in your FileMaker custom app.**
  - Otherwise, the AI Model Server must dynamically load and unload them
- **Do not enable token usage logging in production; it will slow down the embedding models.**

Background

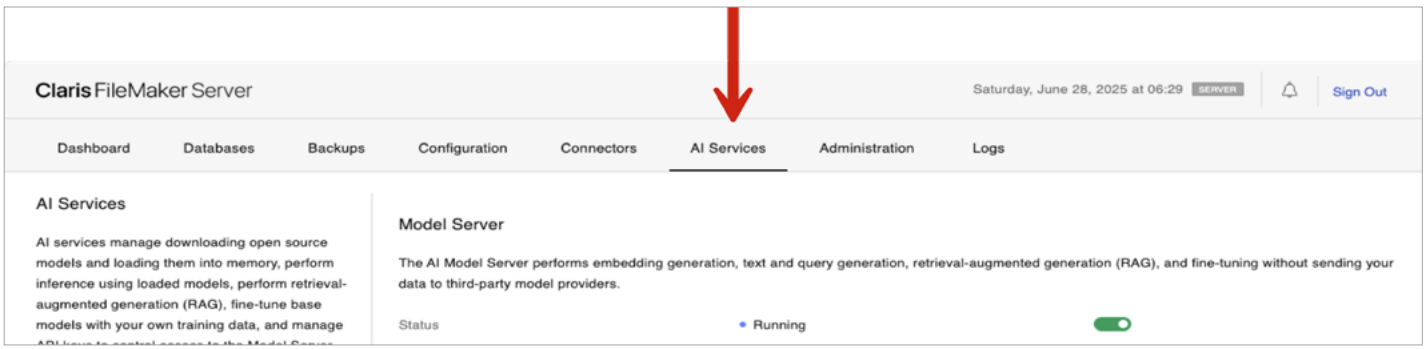
FileMaker Server 2024 (21.0) introduced an AI Model Server component that allows you to run your own local language models (LLMs) for text and image embeddings.

The AI Model Server is built on top of a Python-based microservice that you deploy **separately** from FileMaker Server, based on a set of install scripts in the FileMaker Server Tools folder.



This Community Live webinar from November 2024 covers a complete walkthrough of that installation process.

FileMaker Server 2025 (22.0) makes this installation and configuration process more intuitive by wrapping it in an installer and providing a familiar UI: the FileMaker Server Admin Console.



However, we recommend deploying the AI Model Server on **its own physical or virtual machine** with its own set of resources geared for delivering the best possible AI integration experience and performance.

Think of a FileMaker AI Model Server like a FileMaker WebDirect secondary machine— both separate specific workloads from your main database server. The same logic drives both decisions. Only the setup process differs.

An AI Model Server requires different resources than FileMaker Server. This guide explains these differences and essential concepts.

When hosting your own AI Model Server for text generation, the primary resource utilized is the GPU (Graphics Processing Unit), which handles the core model computations and requires significant VRAM (Video RAM) to store model weights and activations. CPU and system RAM are also used for tasks like tokenization, API orchestration, and caching, but typically aren't the bottleneck unless the model is running on the CPU or is heavily quantized. Disk I/O is mostly relevant during model loading or logging, and network bandwidth supports incoming API requests and outgoing responses.

If you’re using Retrieval-Augmented Generation (RAG), additional resources like vector databases and external storage are also utilized, increasing demand on CPU, RAM, and I/O. Overall, GPU utilization dominates during active text generation, while other system components play supporting roles in managing requests and data flow.

Concepts

When you start your AI journey and are considering running your own AI Model Server, the jargon and terminology may be daunting. In the next sections, we will cut through all of that and give you enough of the basics so that you will feel empowered and not deterred by any of it.

Types of models and FileMaker features

FileMaker Server 2025 supports three types of locally deployed inference models

- Text embedding models
- Image embedding models
- Text generation models

The first two are already supported in FileMaker Server 2024. Support for text generation models is new in FileMaker Server 2025 and enables a slew of new functionality. This table clarifies which FileMaker features utilize specific AI models, helping you determine necessary models and corresponding server resources.

	Text Embedding	Image Embedding	Text Generation
Script Steps			
Fine-Tune Model			✓
Generate Response from Model			✓
Insert Embedding	✓	✓	
Insert Embedding in Found Set	✓	✓	
Perform Find by Natural Language			✓
Perform RAG Action	✓ Add Data		✓ Send Prompt
Perform Semantic Search	✓		
Perform SQL Query by Natural Language			✓
Functions			
GetModelAttributes()	✓	✓	✓
GetEmbedding()	✓	✓	
GetRAGSpaceInfo()	✓		✓

The **Configure AI Account, Configure Machine Learning Model, Configure Prompt Template, Configure RAG Account, Configure Regression Model, Set AI Call Logging** script steps do not make calls to the AI Model Server.

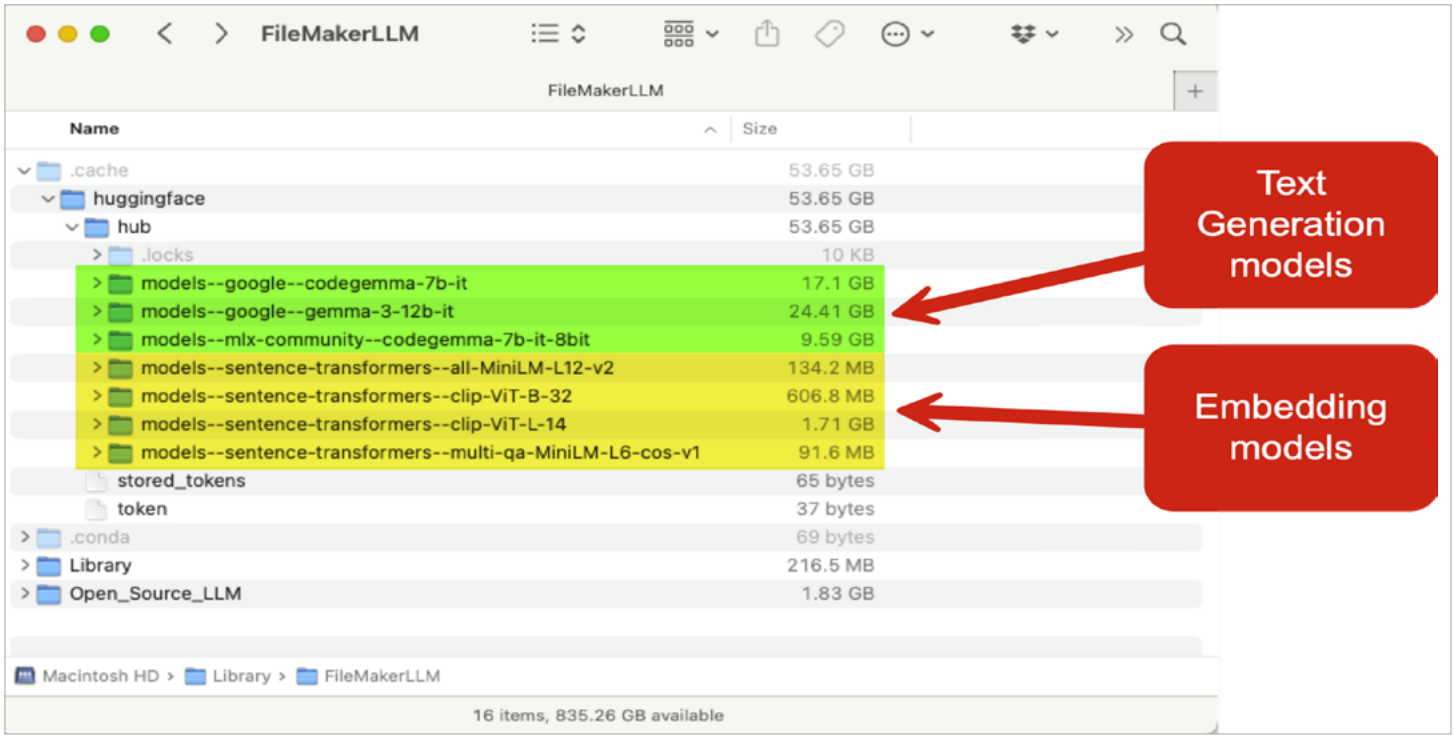
Picking the proper infrastructure for text generation workloads is the most crucial aspect of setting up a Claris AI Model Server.

Here is why.

Model size and memory

Text generation models need more server resources than embedding models because they're larger and require more processing power.

Jumping ahead a bit by showing you where the FileMaker AI Model Server stores the downloaded models, the size difference between embedding models and text generation model sizes is substantial:



And this matters because a model's size directly relates to how much memory it will need to use, which is a **multiple** of its size.

The size of the model is determined by its number of parameters. A model's parameters are the learned weights that determine its behavior - more parameters generally mean it can generate higher-quality outputs.

In the screenshot above, **codegemma-7b-it** has seven billion parameters and **gemma-3-12b-it** has twelve billion.

Note: the "it" in the name stands for instruction-tuned which means it has undergone additional training to enhance its ability to follow instructions and complete tasks as directed by the user.

Quantized models

You will have noticed that there are two **codegemma-7b-it** models highlighted in that screenshot, and the **codegemma-7b-it-8bit** variant is much smaller, about half the size of the other. That is because of quantization.

A quantized model is a machine model where the precision of its parameters has been reduced from, for example, 16-bit floating point numbers (FP16) to lower-precision formats such as 8-bit integers.

Models often come in various degrees of quantization to reduce their size, reduce computational costs, and to improve their speed at the expense of response quality. An 8-bit model will be **roughly** half the size of the base model at 16-bit. A 4-bit model will be approximately a quarter size of the original. The relationship between model parameters and quantization is fundamentally about the trade-off between model capability and computational efficiency based on the server's resources.

So, when you want to know the size of a model and have a rough indication of the amount of memory it will require, you can use this formula:

Number of parameters × Bits per parameter ÷ 8 = model size in bytes

7 billion parameters x 16-bits precision ÷ 8 = 14,000,000,000 bytes = ~ 14GB for the base **codegemma-7b-it** model.

7 billion parameters x 8-bits precision ÷ 8 = 7,000,000,000 bytes = ~ 7GB for the quantized **codegemma-7b-it-8bit** model.

This, however, is just the amount of memory that the model itself will consume, operating the model will require two or three times that amount.

The entire model loads into memory, if it fits, and will then work at its optimum level. In addition to the model itself, extra memory is used for all the activity that interacts with it: the context buffer for the conversation prompt and history, output buffers and key-value cache (KV cache) for each token in the context. This cache will add up on larger models and goes up significantly with the number of users that you will have interacting with the model.

In addition to the model size and the extra memory needed to operate it under load, when you use the AI Model Server's Retrieval Augmented Generation (RAG) feature to feed your own data to a model, that RAG data is also loaded into memory.

In short: the memory usage during text generation is more complex than just loading the model. Allow for sufficient overhead in your available memory over and beyond the size of the model.

The FileMaker AI Model Server can load **one of each model**: one text embedding model, one image embedding model and one text generation model. Keep that in mind to estimate the total memory size you will need across all models you want to have available.

GPU vs. CPU and VRAM vs. RAM

AI models perform best on Graphics Processing Units (GPUs) due to their architecture. While Central Processing Units (CPUs) typically have 4 to 16 cores optimized for sequential tasks, modern GPUs feature hundreds to thousands of cores designed for parallel operations. This parallel processing capability is crucial for the massive matrix multiplications at the core of AI neural network computations, making GPUs significantly more efficient for these tasks.

AI models can work on machines that only have CPUs, Specifically, models stored in Generic GPT Unified Format (GGUF) are optimized for resource constrained environments and CPU-based workloads. While AI models will run on CPU, performance may be degraded. The Claris FileMaker AI Model Server does not support GGUF models.

Graphics Processing Units (GPUs) have their own memory (Video RAM or VRAM) that is usually not shared with the RAM used by the CPUs. For instance, these are the specs of the NVIDIA RTX 4090 graphics card.

	RTX 4090
NVIDIA CUDA Cores	16384
Memory Size	24 GB

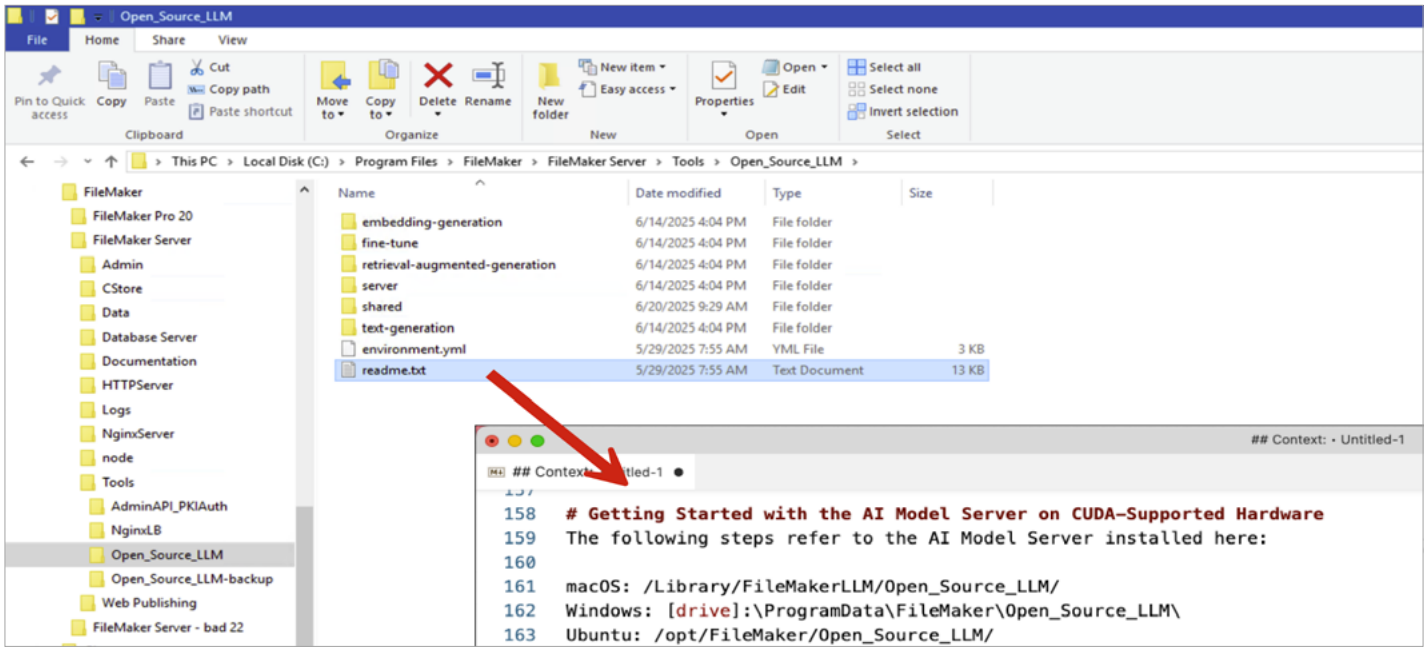
VRAM is the essential memory for AI models. Since VRAM on discrete GPU hardware is a significant limitation, frameworks like CUDA and MLX offer distinct approaches to memory management, including strategies for swapping data between VRAM and RAM. The choice between these frameworks will influence the specific setup tasks required to optimize your system's performance.

CUDA – for Windows and Ubuntu

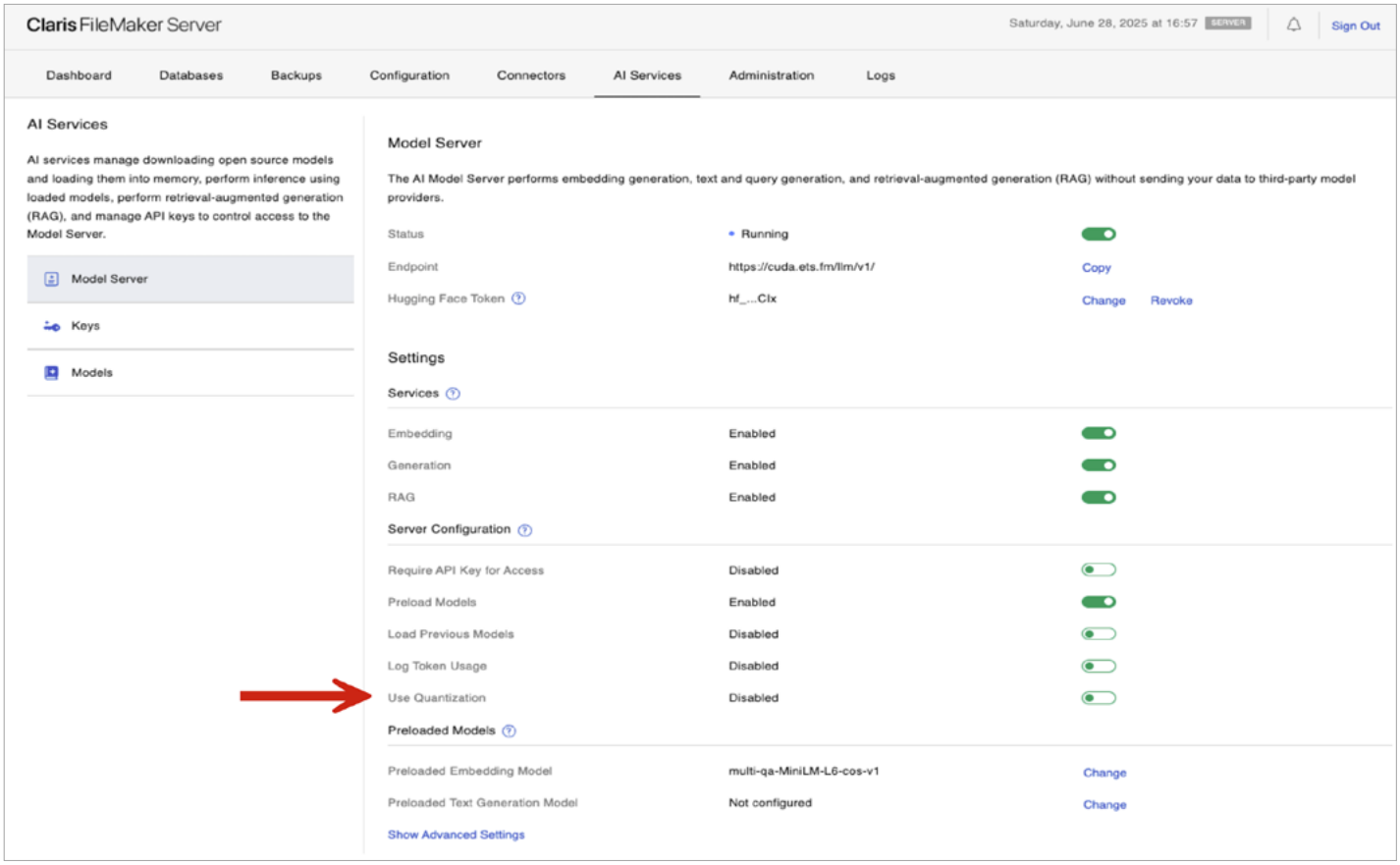
CUDA (Compute Unified Device Architecture) is NVIDIA's framework that lets software use GPU power for general computing, not just graphics.

CUDA works only on NVIDIA hardware, which has separate VRAM from your server's main RAM. The AI Model Server must explicitly copy data between CPU RAM and GPU VRAM, which creates a performance cost. The bigger your VRAM, the less copying happens.

If you have NVIDIA hardware in your server, then follow these instructions in the **`readme.txt`** file to enable it.



Enabling CUDA enables an extra feature in the Claris AI Model Server where you can use quantization.

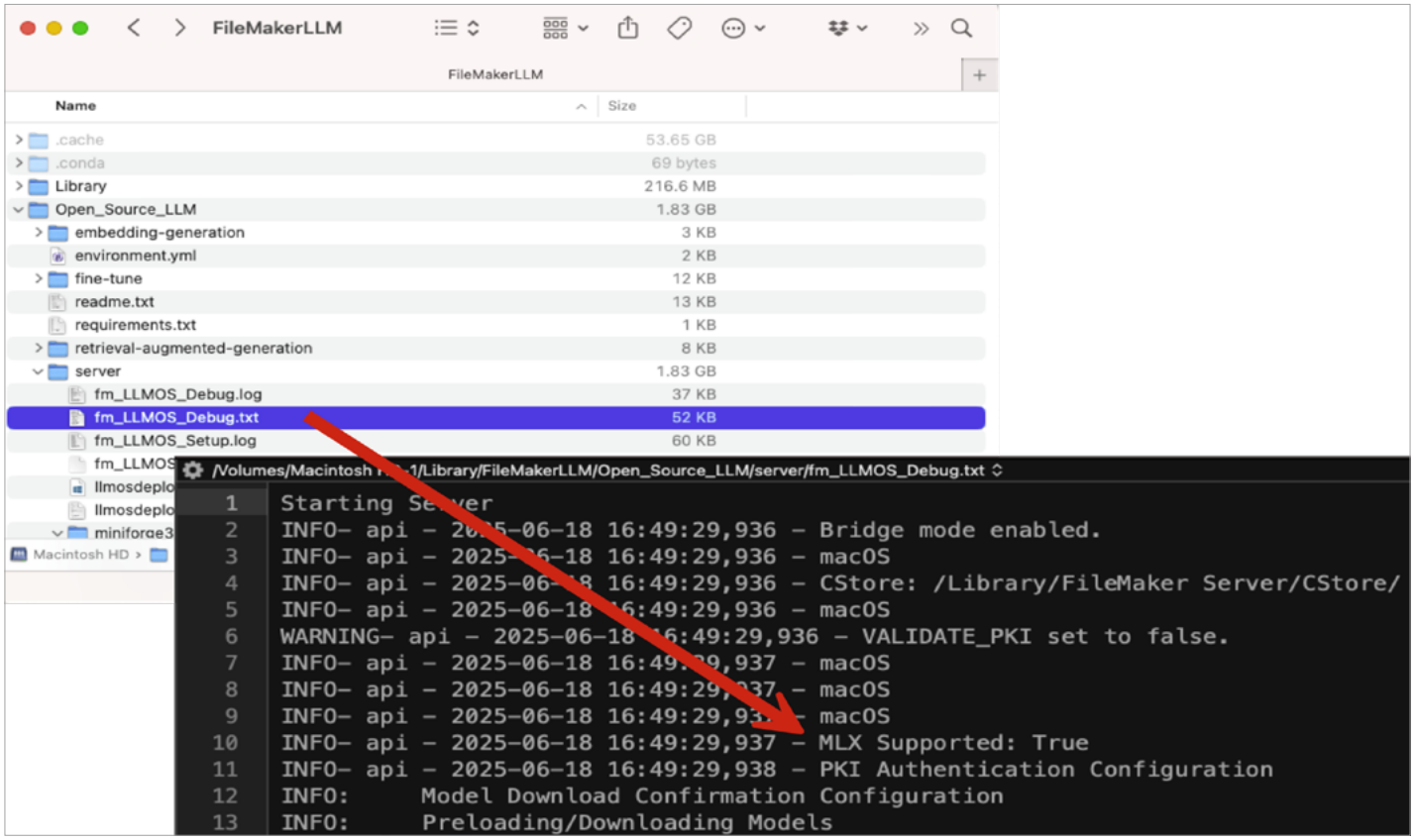


This **Use Quantization** feature on CUDA supported hardware enables 8-bit quantization on full-sized 16-bit models as they are loaded, to reduce the memory impact and increase speed. Studies have shown that the performance impact of this on-the-fly quantization is minimal. Note that this only has an effect on full-size models. If you start with an 8-bit model then no (additional) quantization will happen.

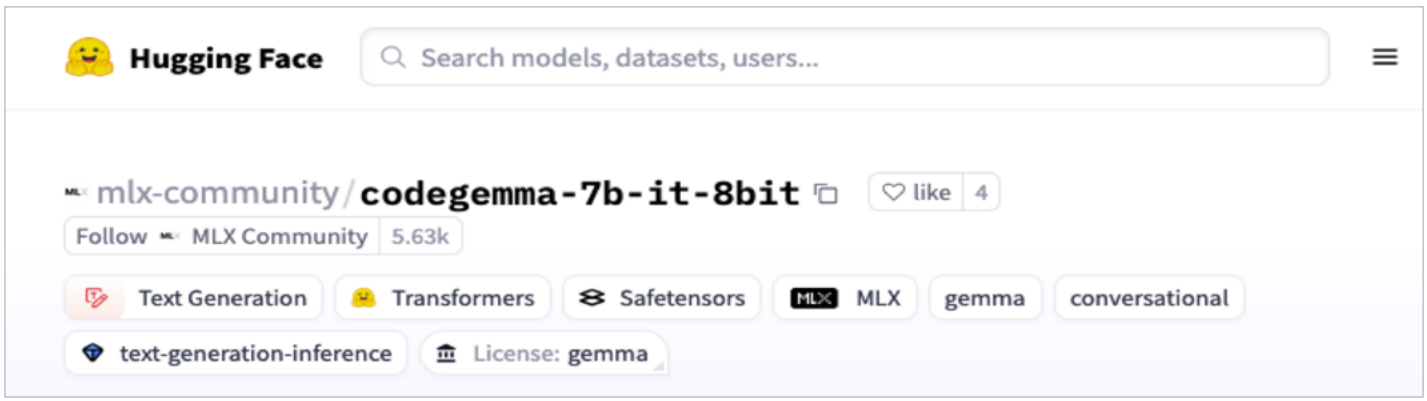
Apple silicon

MLX is Apple's machine learning framework designed for the memory architecture of Apple silicon. Unlike CUDA, MLX uses unified memory where CPU and GPU share the same memory pool. This eliminates the copying overhead between separate memory spaces.

If you're running on Apple silicon, MLX provides better memory efficiency since models can access the full system memory without transfers. When you install the AI Model Server on a Mac with Apple silicon (M-series chips), the AI Model Server installer will automatically enable support for MLX, which you can confirm in the **`fm_LLMOS_Debug.txt`** file.

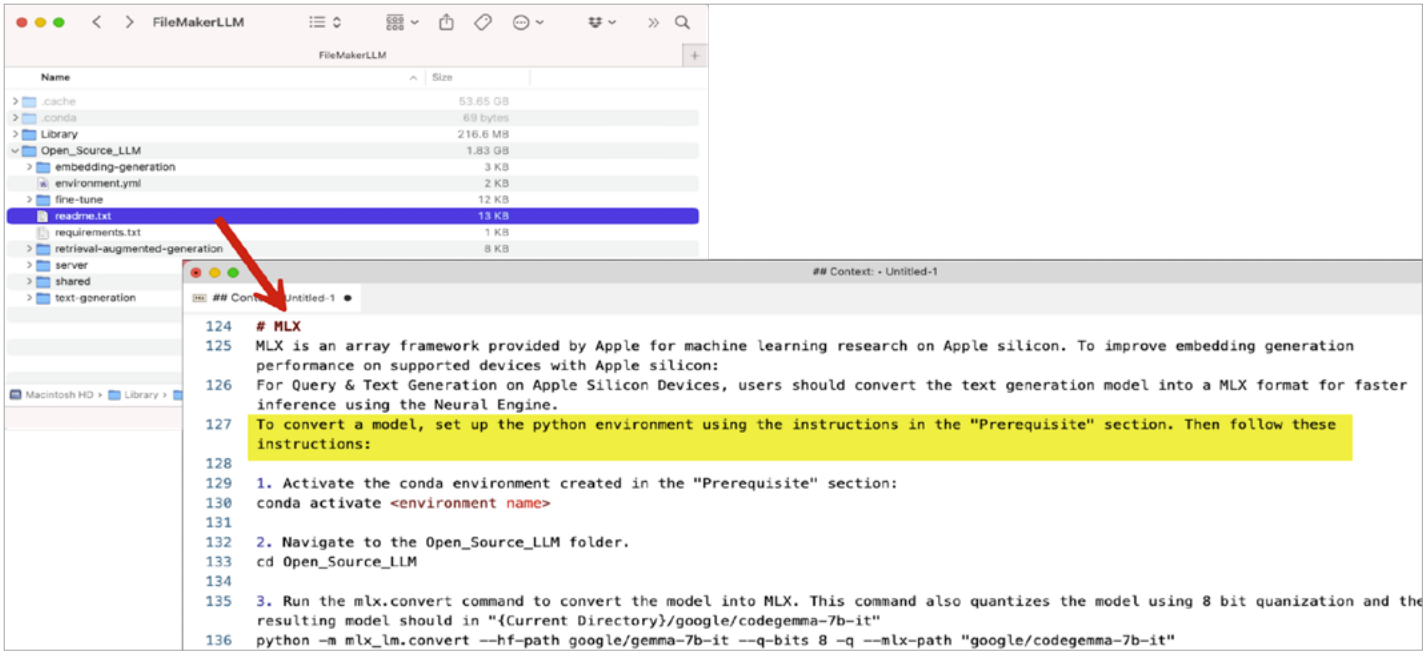


However, for best results you should download a variant of your desired model that has been converted for MLX, like the **mlx-community/codegemma-7b-it-8bit model** below.



There is a distinct performance benefit when using MLX-specific models. The AI Model Server will work properly without MLX-optimized models, but it will be slower.

You can also convert any model to MLX yourself. The readme.txt file mentioned earlier contains the necessary information on how to do this.



When you go through the conversion process you can also quantize the model to reduce its size. Do keep in mind that you will lose some performance and quality when doing so.

Fine-tuning a model

The FileMaker feature to fine-tune a model is only available on Apple silicon (M-series chips), and we expect that tuning an MLX-optimized base model is the most performant scenario.

```
7
8     #  train on table
9     Fine-Tune Model [ Account Name: _AI_account_22 ; Base Model: $model ; Training Data: Table ; “BOOK” ;
      Completion Field: BOOK::summary ; Response Target: $response ]
10
```

Suitable hardware

Armed with everything you have learned up to this point, we can now have a fresh look at the system requirements needed to run your AI Model Server.

Here are some notes on the **Minimum** Requirements. Remember these are specs for a dedicated AI Model Server machine. This is in addition to your main FileMaker Server database-engine machine.

Also note that these minimum requirements are insufficient for larger models running at full precision and with relevant levels of user load. These minimum specs are sufficient to get you going in early exploration of your AI journey, but are unlikely to be sufficient once you are ready for production loads. This is clearly stated with the caveat that it is very hard to anticipate what the production loads will look like across the broad spectrum of custom apps built with the Claris FileMaker platform.

Since most of us are new to AI, starting with low-cost AI Model Servers makes a lot of sense. But view it as a learning exercise that will enable us to more accurately predict what production hardware should look like.

	Windows & Ubuntu	macOS
Text / Image Embedding Only	**CPU:** Intel Xeon or AMD Zen 4, 5 series, with AVX2, or AVX512 **RAM:** 16 GB	**CPU:** Apple silicon **RAM:** 16 GB
Text Generation	**CUDA GPU required** **CPU:** Intel Xeon or AMD Zen 4, 5 series, with AVX2, or AVX512 **GPU:** Nvidia RTX 4090 or Nvidia A10, with combined VRAM of 32 GB or more **VRAM:** 32 GB Depends on size and quantization of the model	**CPU:** Apple silicon Ultra **GPU:** included **(Unified) RAM:** 64 GB
Fine Tuning	Not supported	**CPU:** Apple silicon Ultra **RAM:** 64 GB

Of the potential bottlenecks--memory, number of processor cores, processing power, disk size and I/O, and networking throughput, the most likely constraint is going to be memory. With insufficient memory the models you want to work with may not work properly or may cause instability in your deployment. Memory is usually the limiting factor in AI workloads.

The overall speed of working with AI models, the most tangible expression of your user's experience, is measured in how many tokens per second can get processed.

A token is a chunk of text. Humans read at roughly 200-300 words per minute, which translates to about 4-6 tokens per second (since tokens are often sub-words). This gives us a baseline for what feels natural. While expectations will vary per use case, trying to achieve 15+ tokens per second is considered adequate as it matches what people can process and will feel responsive.

Memory

macOS

You learned earlier in the document that a Mac with Apple silicon has unified memory that can easily be shared between the CPU cores for normal operations and the GPUs cores for AI work.

The amount of memory available in Apple silicon is directly tied to the other resources such as number of CPU and GPU cores and to memory bandwidth (how fast data can move between memory and processors).

Simplifying the choices a bit, this is what your options are for M4 machines:

M4 (Base): 8-10 CPU cores, 8-10 GPU cores, up to 24GB unified memory

M4 Pro: 12-14 CPU cores, 16-20 GPU cores, up to 64GB unified memory

M4 Max: 14-16 CPU cores, 32-40 GPU cores, up to 128GB unified memory

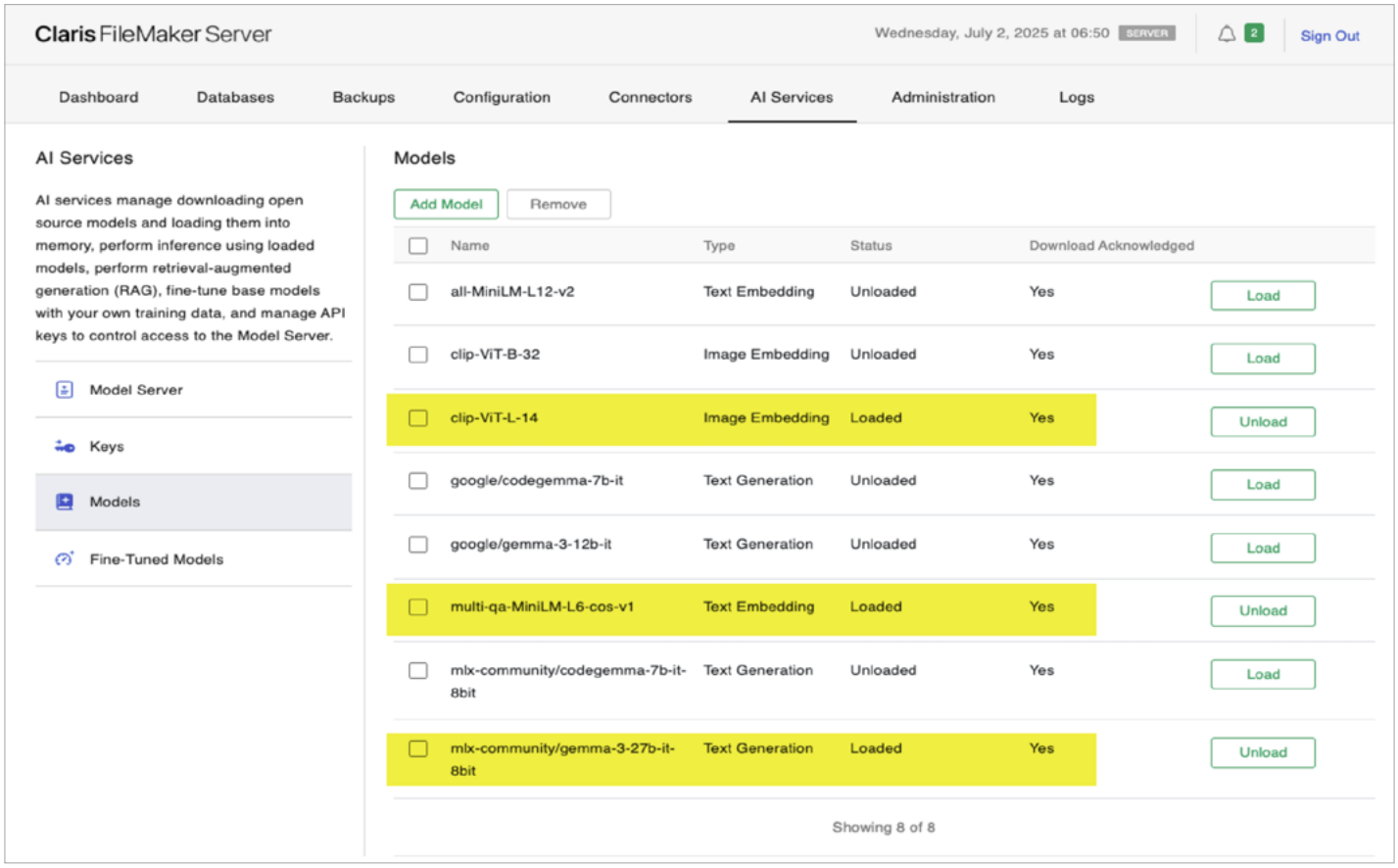
M3 Ultra: (which is a setup with dual Max chips connected), ~28-32 CPU cores, ~60-80GPU cores, up to 512GB unified memory.

There isn't just more memory in the top tier models, they can access that memory much faster. This is crucial for AI because more memory bandwidth helps with:

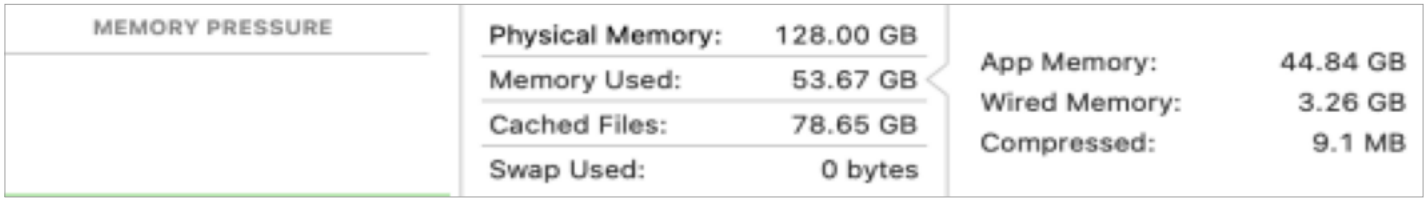
- Larger models, these require streaming more data from memory
- Reducing the bottleneck between memory and compute resources
- Better utilization of the additional GPU cores

The higher-end machines can achieve two or three times as many tokens per second as the lower tiers (10-20 for a base M4 vs up to 30-60 for an M3 Ultra).

As a practical example, a Mac Studio M4 Max, deployed as a dedicated AI Model Server has these three models loaded. The biggest of course is the text generation model since it is a 27-billion parameter model but quantized to 8-bit to help reduce its memory footprint.



And it's memory use at rest (not under load) shows just over 50GB of memory in use.



The larger the model the more memory you will need.

Windows/Unbuntu

For Windows and Linux, the size of the VRAM matters, the larger the model (and the associated accuracy) and the heavier the load, the more VRAM is needed.

In addition to the size of the model itself (which ideally can be loaded completely in memory), there is also the memory required by the interactions with it (buffers, cache, and so on.)

RAG

If you use Retrieval Augmented Generation (RAG) then you must factor in the size of the RAG data. This is wholly loaded in memory under ideal performance circumstances. The more data you feed to your RAG space the more memory will be required.

Processing Power

For Windows and Linux, AVX2 and AVX512 matter because these are CPU instruction set extensions that dramatically accelerate AI workloads by enabling parallel processing of multiple data elements in a single instruction. From earlier in the document you will have picked up that GPUs are all about parallel processing. The CPU should have some of that same capability especially on Windows and Linux since it is more likely that the CPU will be more involved in the actual AI work if the GPU has insufficient memory.

Apple silicon has GPUs built-in so there are fewer choices to be made there. The main choice is the amount of memory. Prioritize the memory and accept the processing power that comes with that.

Disk Size and Speed

The Claris AI Model Server downloads models to the hard disk. As seen in the screenshot earlier in this document, that can add up to a significant amount of disk space.

Given the size of the files, especially the text generation models, when the AI Model Server needs to load the model, having fast disk I/O matters, but memory is more important. If your server does not have sufficient memory, that is going to degrade performance more than poor disk I/O.

When you install the Claris AI Model Server on Ubuntu, make sure that you configure Ubuntu to use a swap file. If you use the Claris FileMaker Server installer, you will be prompted to do so.

Models are saved to these locations when downloaded:

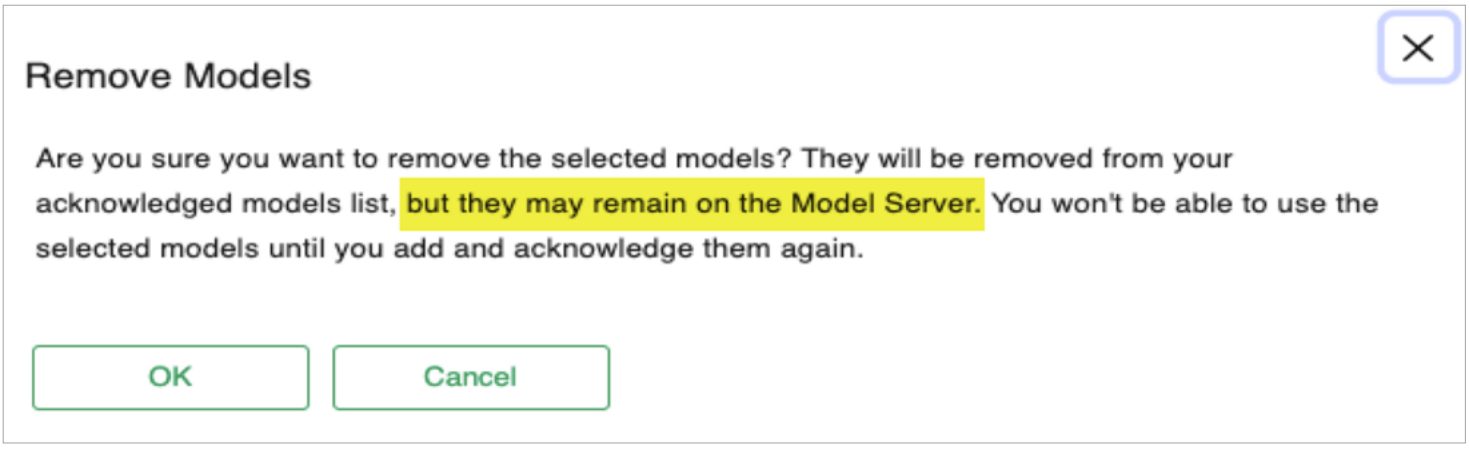
Windows: C:\Windows\System32\config\systemprofile\.cache\huggingface\hub

macOS: /Library/FileMakerLLM/.cache/huggingface/hub/

Ubuntu: /opt/FileMaker/.cache/huggingface/hub/

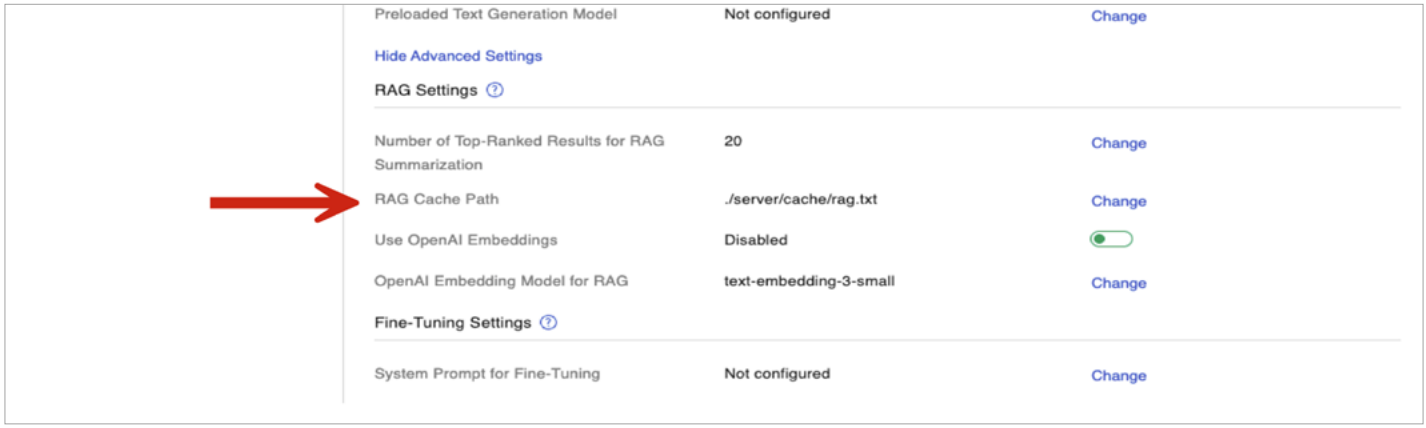
Be mindful of the amount of disk space that is consumed by these models. In the screenshot early in this document, there is a total of 55 GB being consumed.

When you unload and remove a model; the downloaded model itself is NOT deleted from your hard drive:



As you experiment with different model sizes, keep this in mind.

Any data that you add through Retrieval Augmented Generation is also stored on the hard disk:



And finally, enabling the AI Services installs the Python microservice (miniforge, all dependencies and libraries) and that takes about 2 GB of disk space.

Usage Patterns

Your AI work will likely be characterized by short but very intense periods of very high activity and longer periods of relatively steady less intense usage.

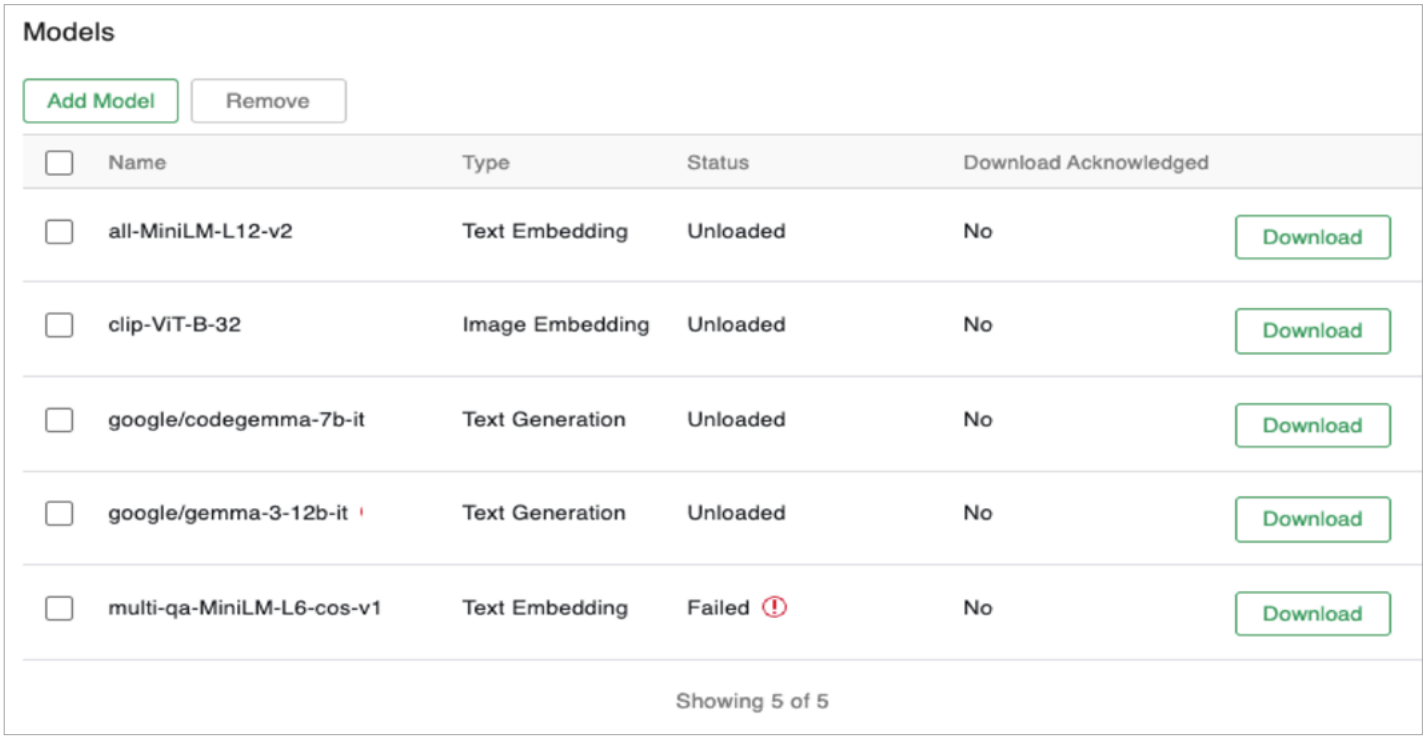
Those high-activity demands will come from:

- Generating initial bulk text and picture embeddings. It is likely that you will be doing this more than once as you work through finding the right model that provides the best speed and accuracy combination for your use cases.
- Adding RAG data

The higher-end machines can achieve two or three times as many tokens per second as the lower tiers (10-20 for a base M4 vs up to 30-60 for an M3 Ultra).be required.

Default models

Claris offers several models as default choices via the AI Services admin console panel. Two text embedding models, one image embedding model and two text generation models.

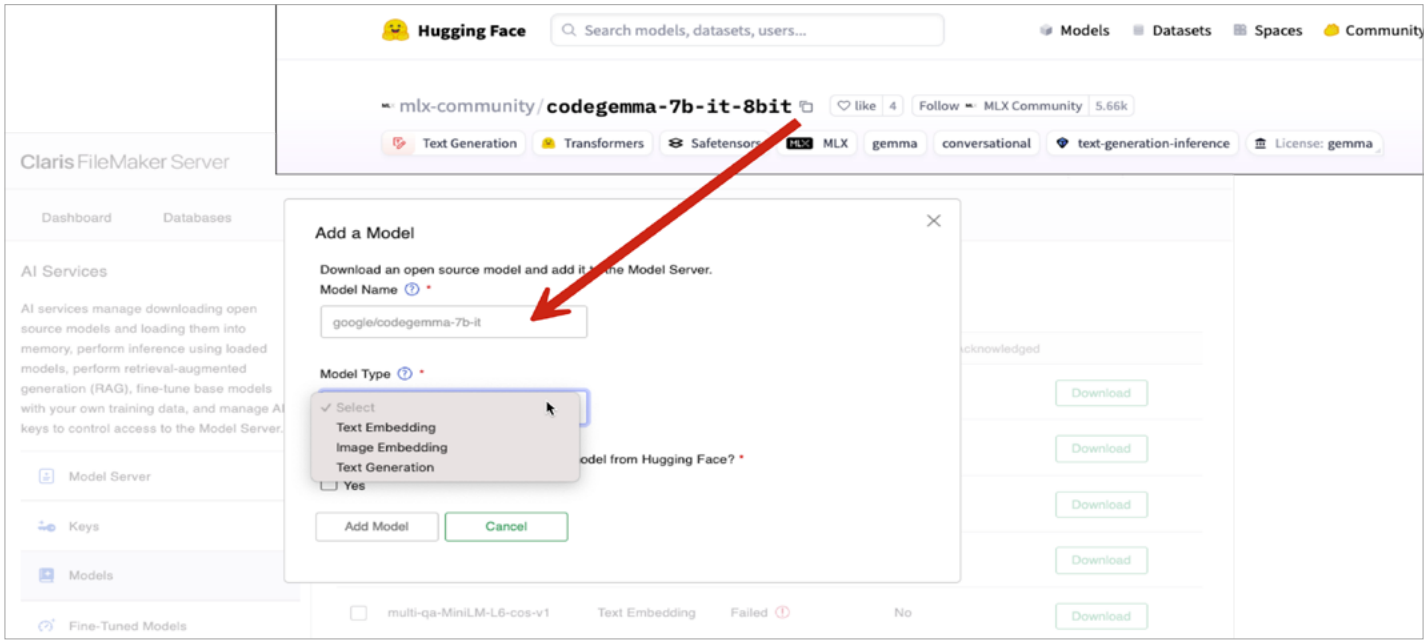


Note that none of these models is downloaded by default. You need to select which ones you want to use. When you download one it will also get loaded.

The Claris AI Model Server can load only one of each type. So if you download two models of the same type, the last one downloaded will be loaded and the previously downloaded one will be unloaded. A model can only be used when it is loaded.

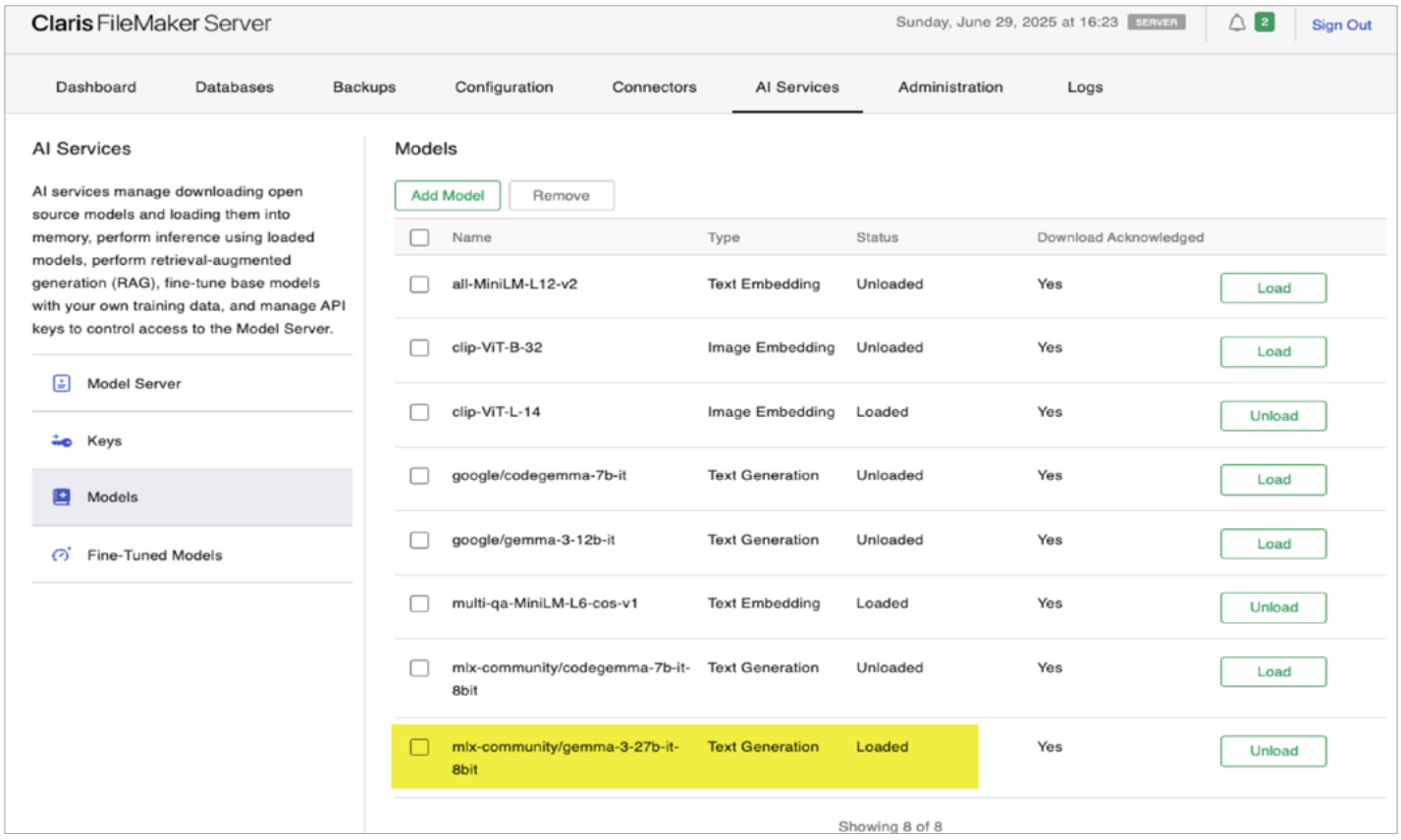
You can add additional models to this list. Huggingface.co is the place to go and you can copy the name of the model, add it to the AI Model Server's models and the model will get downloaded and then loaded.

Models of the same type cannot be used while another model of that type is downloading. This is a known issue at the time of this release.



Only one model of each model type can be loaded at a time. It is very important to understand this. You can configure multiple models of the same type, but only one will be loaded at a time. If your FileMaker solution calls the different models, the Claris AI Model Server will unload one model and load the other. This may have performance implications as users may need to wait for the current process to finish before their model can load.

An example: with the configuration below the AI Server has the **mlx-community/gemma-3-27b-it-8bit** model loaded.



This script will execute the first call without issues. But the second call uses the google/gemma-3-12b-it model, which is available on the server but not loaded:

```
5
6   Generate Response from Model [ Account Name: "LLM" ; Model: "mlx-community/gemma-3-27b-it-8bit" ; User Prompt: Quick agent::Prompt ; Agentic mode ;
    Response: Quick agent::Reponse ; Messages: $$messages ; Save Message History To: $$messages ; Message History Count: 10 ]
7
8   Generate Response from Model [ Account Name: "LLM" ; Model: "google/gemma-3-12b-it" ; User Prompt: Quick agent::Prompt ; Agentic mode ; Response:
    Quick agent::Reponse ; Messages: $$messages ; Save Message History To: $$messages ; Message History Count: 10 ]
9
```

This will cause the Claris AI Server to unload **mlx-community/gemma-3-27b-it-8bit** and load **google/gemma-3-12b-it.**

Given the size of the text generation model, the impact of unloading & loading will be the biggest.

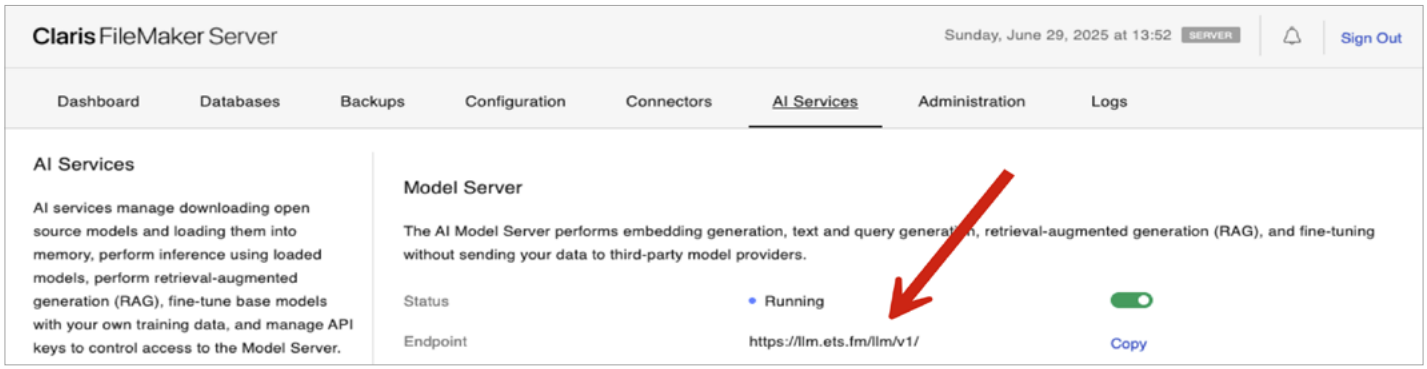
So avoid calls to different models of the same type.

How

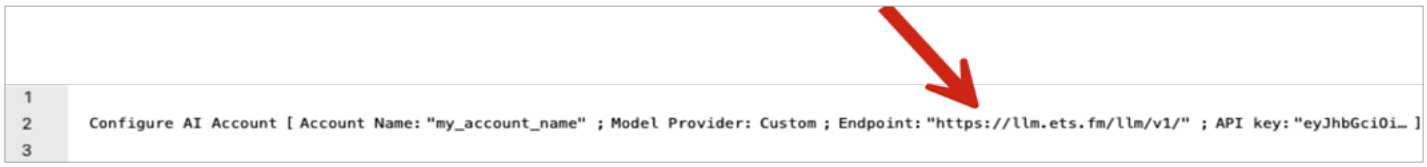
Through the FileMaker Server Installer

You install the Claris AI Model Server, on its own machine with the normal Claris FileMaker Server installer. Choose the option to install the primary server (not the option to install on a FileMaker WebDirect secondary machine).

When the installer finishes, add a custom SSL certificate, because once configured, the Claris AI Model Server is reachable through an HTTPS URL:



Then you use that URL in the **Configure AI Account** script step:

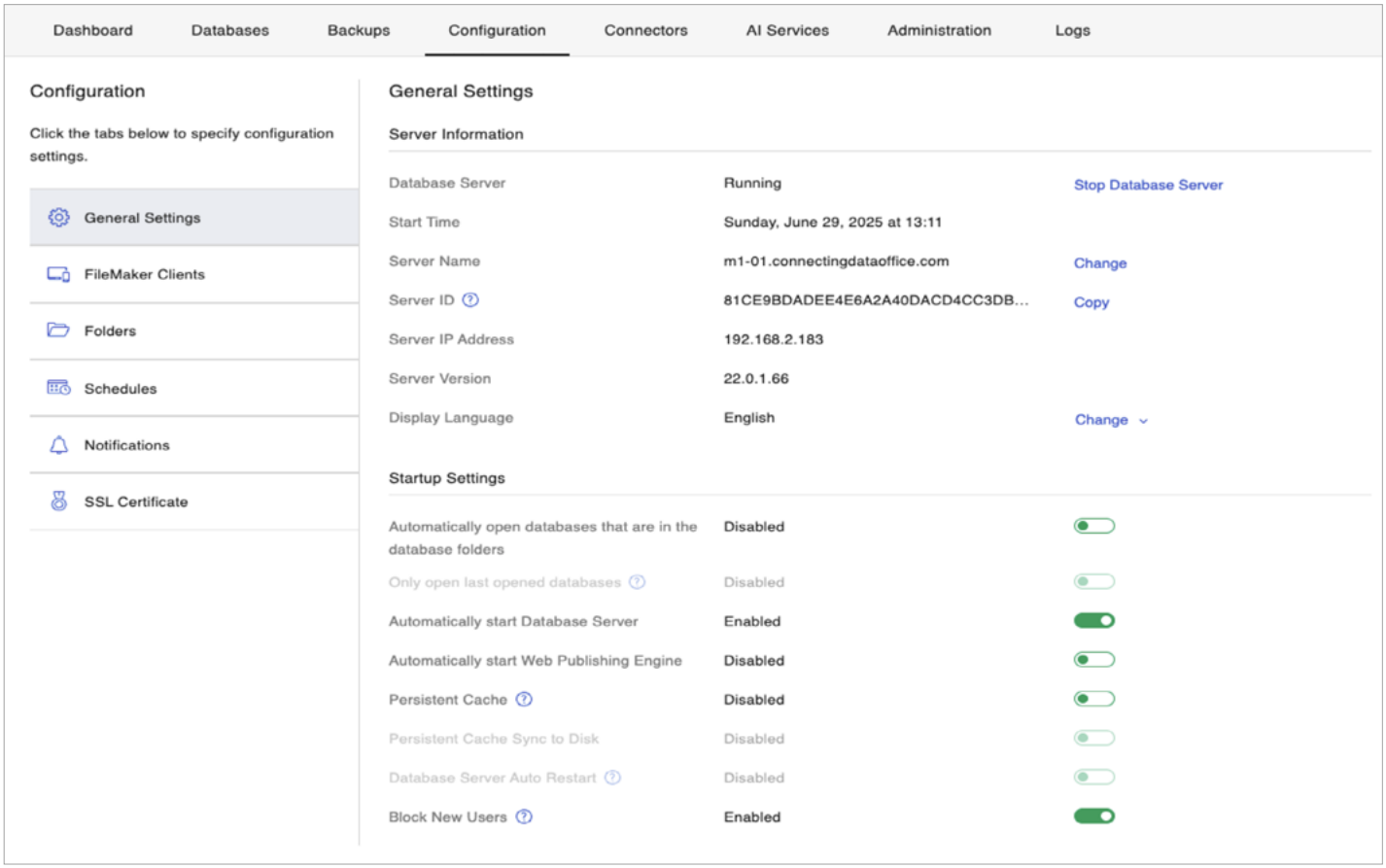


Since you will not use this machine as the Database Server, close and remove the Sample file, if it was installed, and for good measure turn on the feature to **Block New Users**.

The Database Server however needs to be running for the AI Model Server to be functional. So do not turn off the setting to **Automatically Start Database Server**.

Automatically Start Web Publishing Engine should be turned off.

Your **Startup Settings** should look like this:



There is one more crucial setting that needs to be adjusted: the size of the database cache. That setting is not available in the admin console.

From the command line on the server, execute this command:

fmsadmin set serverprefs cachesize=64

This matters because FileMaker Server by default uses a database cache of 512MB, that it monopolizes and does not share with other processes. Switching to the lowest amount allowed (64MB) frees up most of that.

Manually

Fundamentally, the AI Model Server in Claris FileMaker Server 2025 is still a python microservice like the one that is available in FileMaker Server 2024. However, you cannot install it as a standalone entity like you could with FileMaker Server 2024. You need to use the FileMaker Server installer.

Disabling the AI Model Server

On the main FileMaker Server that you use to host your FileMaker files, you can disable theAI Model Server by removing the UI from the Admin Console. That way you can prevent admins or developers from accidentally enabling the AI Model Server and impacting the performance and stability of hosting your custom apps.

You do that by adding the following JSON to a file named ClarisConfig.json:

```
{
 "DisableAIServer": true
}
```

And placing that JSON file in this folder:

On Windows: C:\Program Files\FileMaker\FileMaker Server\Database Server\

On macOS: /Library/FileMaker Server/Database Server/bin/

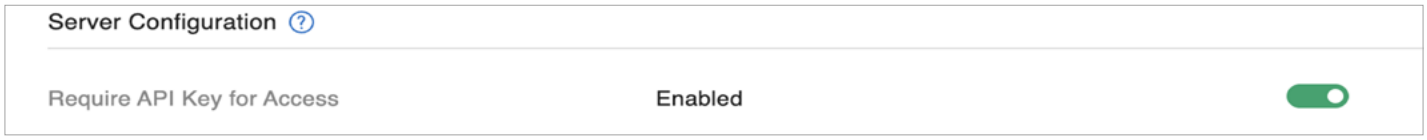
On Ubuntu: /opt/FileMaker/FileMaker Server/Database Server/bin/

On the next FileMaker Server restart the AI Services tab will be removed.

Security Considerations

When running your own FileMaker AI Model Server, it's critical to take proactive steps to secure the AI endpoints and data pipelines. Below are two key areas that require special attention: API access controls and SSL encryption.

RAG: Require API Key for Access



By default, the **Require API Key for Access** setting is disabled, which means the AI Model Server and RAG (Retrieval-Augmented Generation) store are open to unauthenticated access. This can lead to serious data exposure risks, especially if your RAG store includes proprietary or sensitive business information.

We strongly recommend enabling **Require API Key for Access**. This ensures that only authenticated clients can interact with the AI endpoints and query your RAG data.

SSL

Claris FileMaker Server ships with a default self-signed SSL certificate, which enables encrypted communication but is not trusted by browsers or FileMaker clients. As a result, users may encounter warning messages or failed connections unless they manually bypass security checks. This undermines trust and usability, especially in production environments.

Using a valid SSL certificate issued by a trusted Certificate Authority (CA) ensures that client connections are secure and authenticated, eliminating browser and client warnings. It also prevents man-in-the-middle (MITM) attacks by verifying the server's identity during SSL handshakes, something a self-signed certificate cannot do effectively.

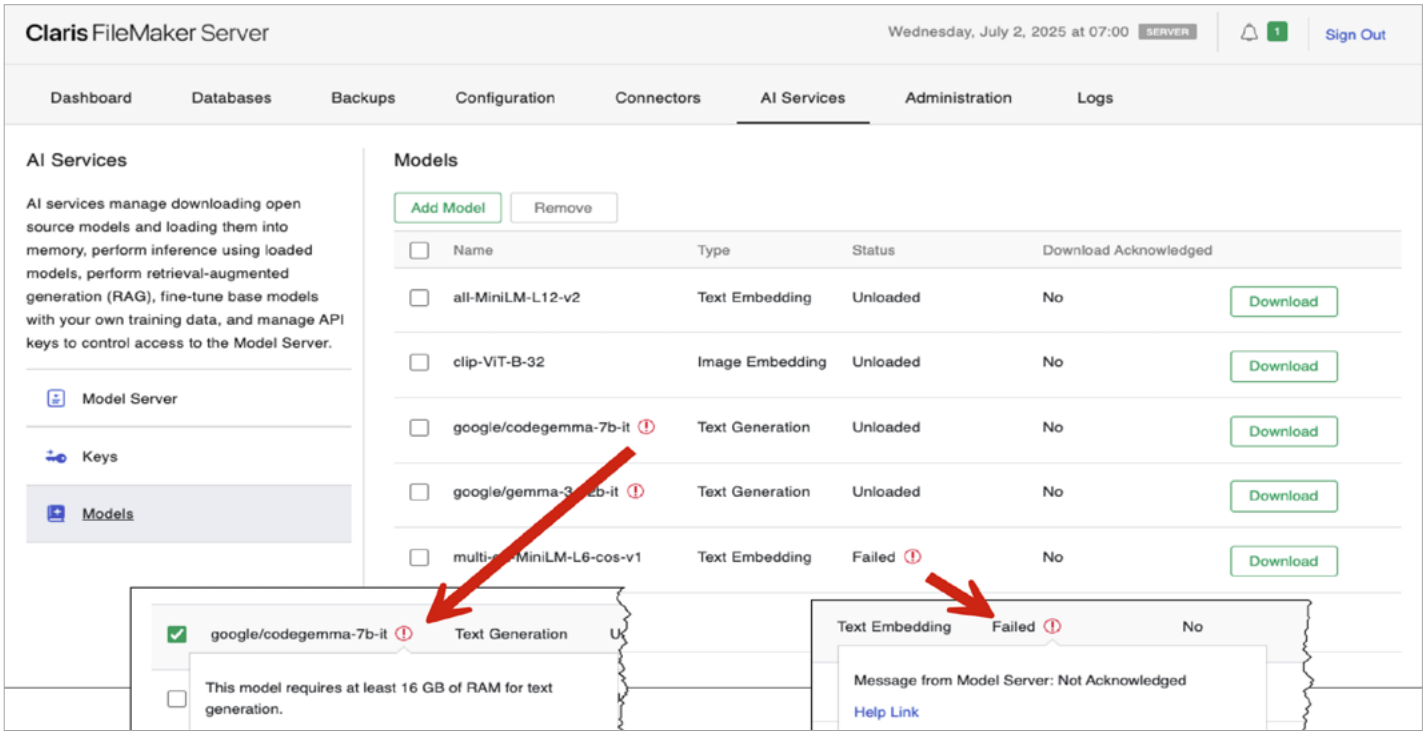
Many integrations such as AI services, webhooks, and third-party APIs require a valid SSL certificate to establish secure communication. Without one, automated tools may reject connections or fail to transmit data securely. Additionally, using a trusted certificate helps meet compliance standards like HIPAA, SOC 2, and GDPR, which require strong encryption and validated server identities.

Troubleshooting

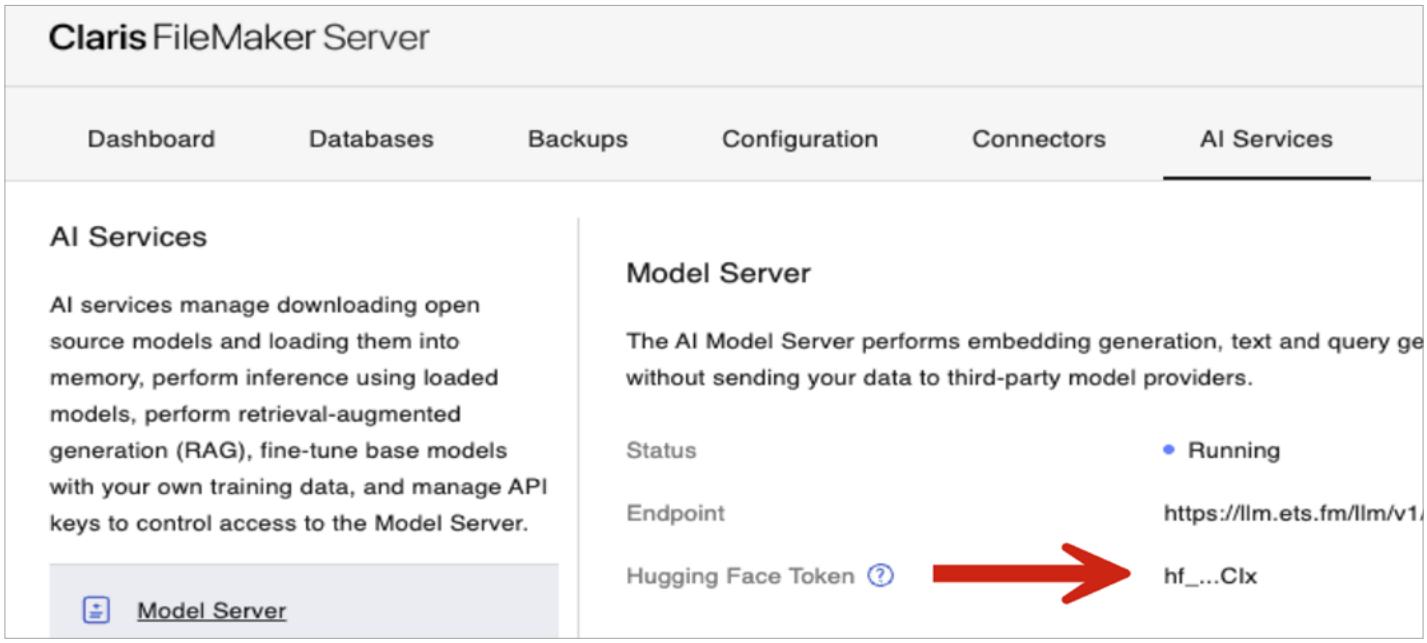
This document does not aim to be an exhaustive guide to troubleshooting an AI Model Server deployment, but we want to arm you with the basic information to help you along.

When you first enable the AI Model Server you may see red exclamation marks next to some models.

These will indicate what the problem is. For example, if the available memory in the machine is not going to allow for smooth operation of the model. Or if you have not yet acknowledged the model for download.



Some models will also require that you have a Hugging Face account (it's free) and that you explicitly requested access to certain models. The Gemma models for instance require that. Once you have your Hugging Face account, generate an API token and add it to the AI Services settings:



There are two logs that can help provide insight into how the AI Model Server is running: fm_LLMOS_Debug.txt and fm_LLMOS_Debug.log.

You will find them in these locations:

Windows: C:\ProgramData\FileMaker\Open_Source_LLM\server\

macOS: /Library/FileMakerLLM/Open_Source_LLM/server/

Ubuntu: /opt/FileMaker/Open_Source_LLM/server/

You may only see one of these two logs depending on whether you have enabled Token Usage logging in the AI Services settings.

Acknowledgements

Authored by:

- Wim Decorte – Soliant Consulting
- Ernest Koe – Proof+Geist

In collaboration with:

- Wade Ju – Claris, an Apple Company
- James Parker – Claris, an Apple Company
- Masao Aoto – Claris, an Apple Company
- Derek Lee – Claris, an Apple Company
- Derek Hwang – Claris, an Apple Company
- Lucy Chen – Claris, an Apple Company
- Ian Jempson – Transforming Digital
- Brian Ouimette – Proof+Geist
- Chris Maddox – Proof+Geist
- Todd Geist – Proof+Geist
- Karl Jreijiri – Soliant Consulting
- Sara Severson – Soliant Consulting
- Steve Lane – Soliant Consulting

Reviewers:

- Ronnie Rios – Claris, an Apple Company
- Agnes Riley – Claris, an Apple Company

© 2025 Claris International Inc. All rights reserved. Claris, FileMaker, FileMaker Pro, and the Claris logo are registered trademarks of Claris International Inc., in the U.S. and other countries and regions. All other trademarks are the property of their respective owners.