AI Gateways: What are they and why should you use one?

What is the purpose of an AI Gateway?

More and more software developers are turning to the use of large shared AI models to add communication, understanding, and analytical intelligence to their applications. Large language models such as GPT, Gemini, Llama, Mixtral, and Claude are among the most frequently used. At the same time, hundreds of new models are constantly being deployed and upgraded by competitors and open-source communities. In addition, use of audio, image, video, and numerical models continue to rise with no end in sight. In order to connect to and use this plethora of AI resources, developers are faced with a constantly evolving landscape of vendors, provisioning, API endpoints, and resource management. AI Gateways are a solution to help simplify and manage AI resources while optimizing speed, security, and privacy.

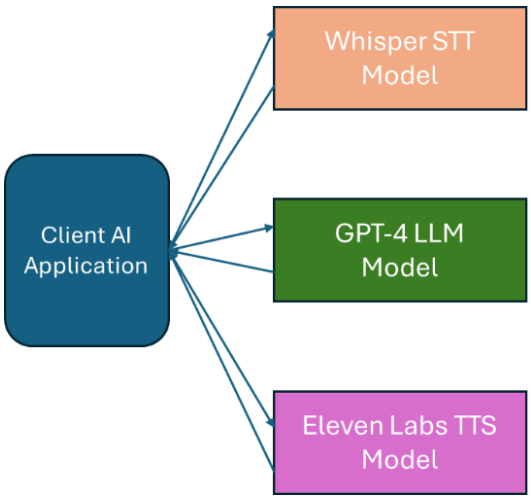
What exactly is an AI Gateway and how does it work?

The purpose of an AI Gateway is to simplify management and use of distributed AI resources, reduce cost, improve performance and reliability, and provide a single unified mechanism for handling data, safety, and security. An AI Gateway serves as a kind of centralized switchboard sitting between your AI applications and distributed AI models. This switchboard receives incoming messages from any number of client applications, routes them to an appropriate model for processing, and communicates the results back to the calling application. AI Gateways may perform a number of other useful functions during this process including message translation, safety validation, resource load balancing, caching, message prioritization, and data aggregation. In the pages that follow we'll look at each of these functions in detail, starting with the basic routing function:

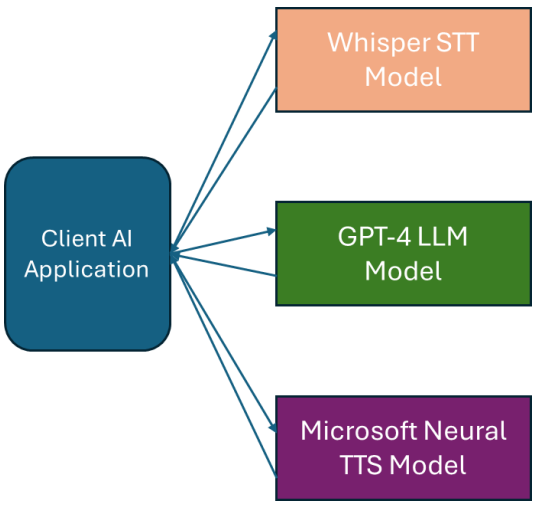
Author: William Klein, bill.klein@y2ai.co, https://y2ai.co

Routing:

The most basic function of an AI Gateway is routing. Routing is the “switchboard” aspect of an AI Gateway and is important whenever an AI request may be served by more than one AI resource. For example, imagine a conversational AI application involving STT (speech to text), LLM (large language model), and TTS (text to speech) models. When a user of the application speaks, their voice is translated to text using STT. The results are sent to an LLM for conversational processing. The output of the LLM is then sent to TTS to create audio of the AI’s voice to be played back to the user. Without an AI Gateway, that flow may look like this:

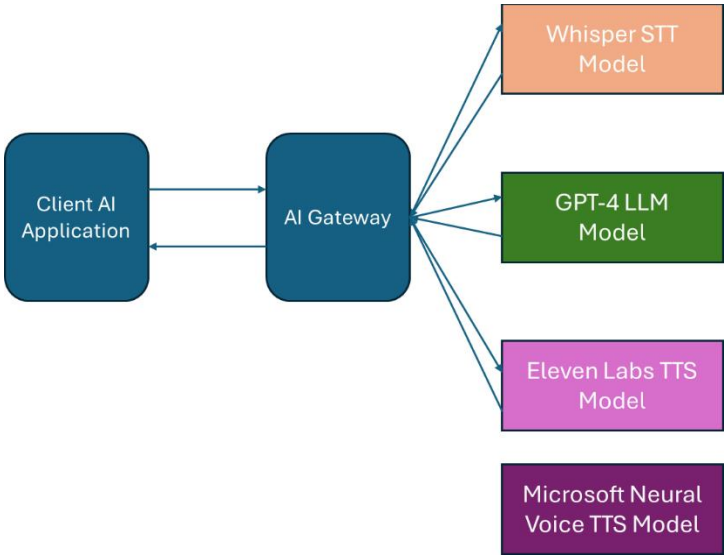


As you can see, the Client AI Application must manage communication with each AI resource independently. And if one of the resources changes, for example, if Microsoft Neural Voice is substituted for Eleven Labs, then the Client AI Application must be updated and redeployed.

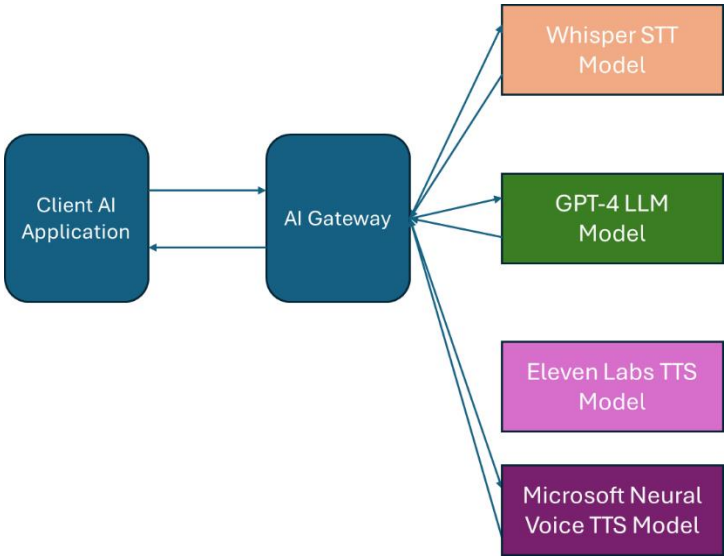


Author: William Klein, bill.klein@y2ai.co, https://y2ai.co

Let's reimagine this scenario using an AI Gateway:



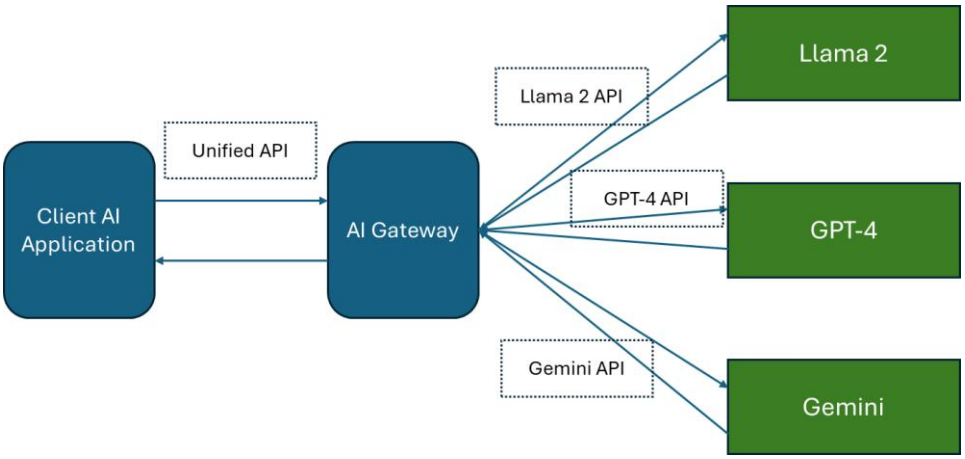
When using an AI Gateway, the client application is no longer directly connected to AI resources. In fact, it only requires a single connection to the AI Gateway. The client application may be coded and deployed independently from the model infrastructure. In addition, changes to AI resource can be automatically handled by the AI Gateway, which simply re-routes incoming calls to the alternate resource, while the client application remains unaffected.



Author: William Klein, bill.klein@y2ai.co, https://y2ai.co

Message Translation:

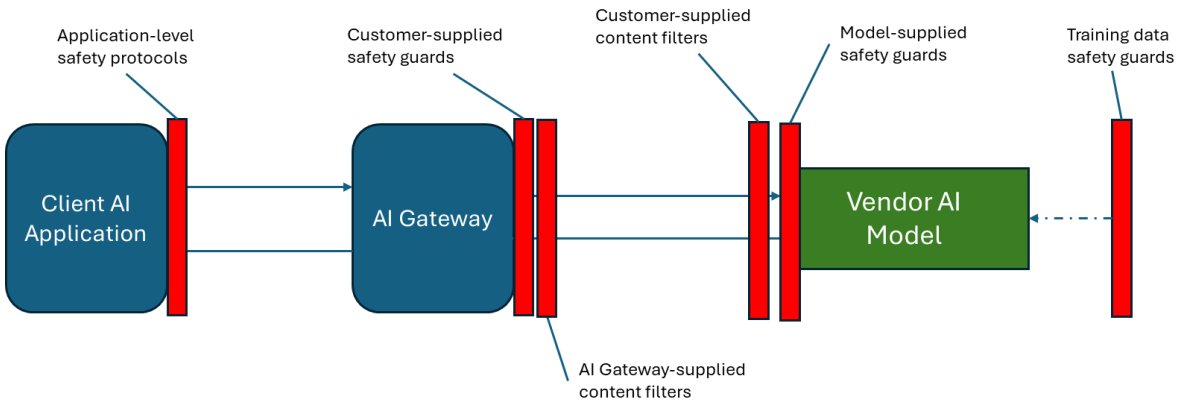
In the prior routing example, we've ignored a critical piece of the puzzle. Not all AI resources speak the same "language." While switching from one TTS model (Eleven Labs) to a different TTS model (Microsoft Neural Voice) might seem like a simple change, in reality the API, call parameters, model parameters, and data formats may be very different between the two. This is where message translation comes in, and the issue is resolved when an AI Gateway supports a single unified API of its own. While each AI Gateway provider may offer a slightly different solution, the general characteristics of the API remain the same. The AI Gateway's API should support all of the major features of all of the models that it connects to. For LLMs, this means the Gateway will publish an API that supports simple completions, chat-based messages, vector embeddings, and function-calling (at a minimum). For other types of models, including video, image, and audio, the API should incorporate all standard operations.



Author: William Klein, bill.klein@y2ai.co, https://y2ai.co

Safety Validation:

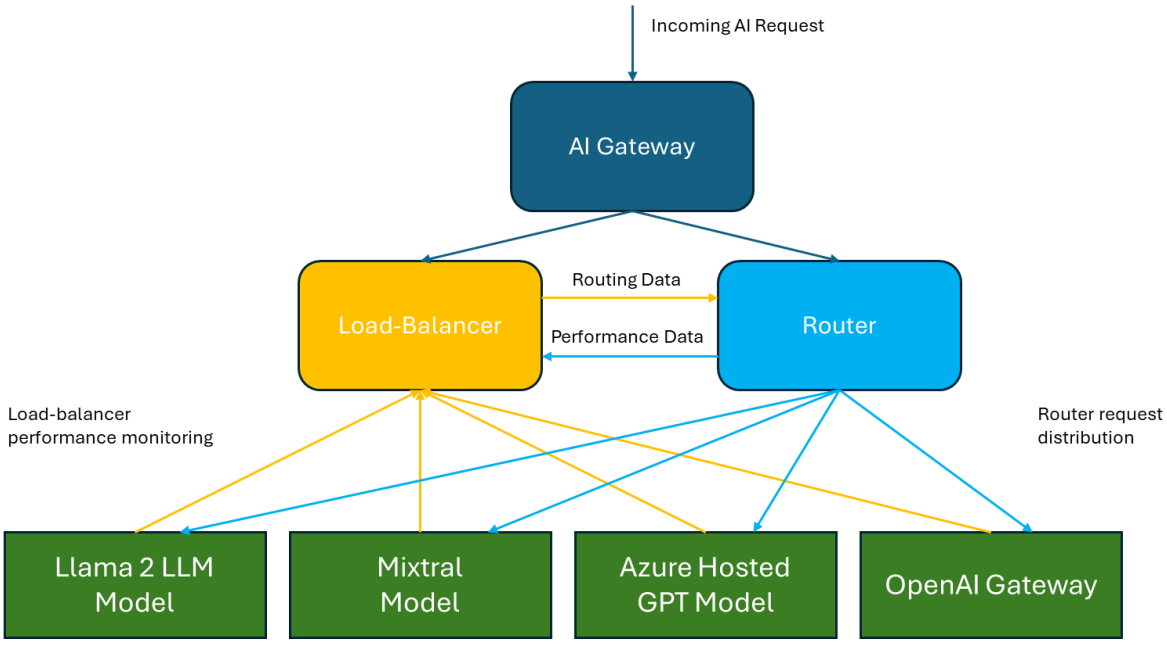
One of the most important elements in AI applications is maintaining the safety of users and protecting them from harmful AI outputs. AI safety is an important concern across the AI industry and AI model vendors often include safety measures as required aspects of their use. Model vendors may also support customer-defined custom AI filters deployed alongside models as extra safety measures. In a multi-model environment, as those supported by AI Gateways, it also makes sense to support application-defined safety protocols within the AI Gateway itself. While no standard exists for AI Gateway-level safety support, AI Gateways may support a combination of built-in filters (for example, to remove or block content inappropriate for children) as well as custom-coded filters supplied by customers. The diagram below shows different levels of AI safety in the AI call chain.



Author: William Klein, bill.klein@y2ai.co, https://y2ai.co

Resource Load Balancing:

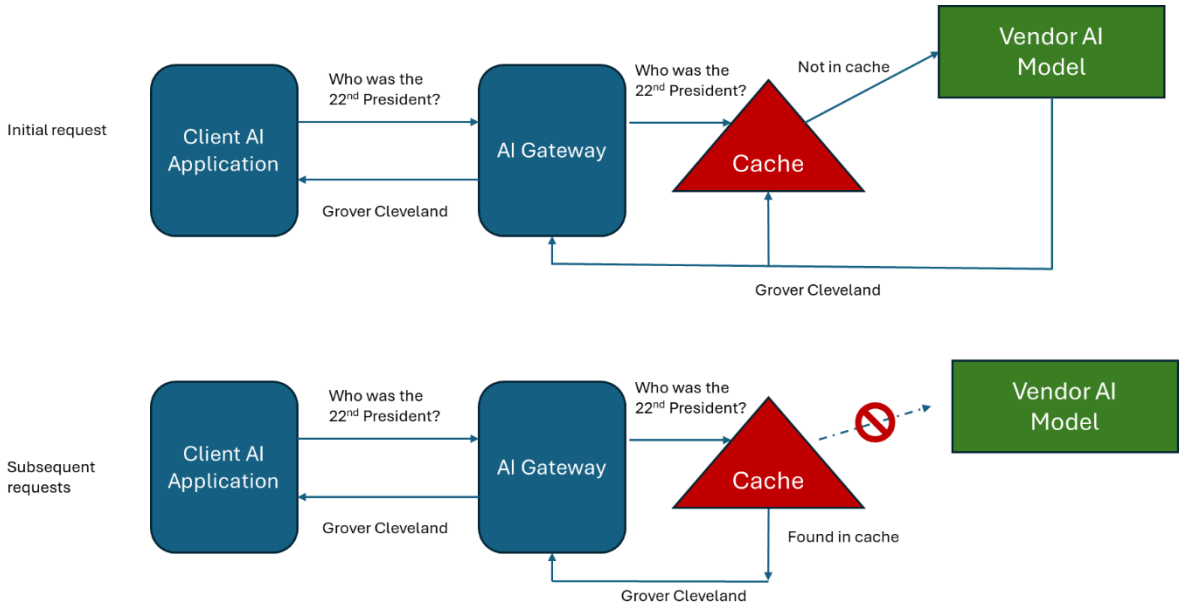
Another important feature of many AI Gateways is the ability for the Gateway to improve overall system performance, minimize latency, and improve operational reliability and up-time through a load-balancing architecture. In general, load-balancing is the idea of distributing work across your resources so that no one resource is overburdened or becomes a bottleneck to performance. A good load-balancer will automatically detect when a resource is overused and will redirect work to other resources. A load-balancer may also automatically detect resource failures, for example if an AI vendor suddenly experiences a service disruption, an AI Gateway can seamlessly redirect AI request to alternate available resources. In the best case, a load-balancer can work cooperatively with the AI Gateway routing capability to distribute resources based on a number of desirable characteristics, such as latency, throughput, cost, and even environmental impact. Basic load-balancers are capable of distributing work across identical resources (e.g., distributing an AI request across multiple Llama 2 instances.) More advanced load-balancers are capable of distributing work across clouds, vendors, and model types. The diagram below illustrates the use of a load-balancer in an AI Gateway:



Author: William Klein, bill.klein@y2ai.co, https://y2ai.co

Caching:

Caching is the ability of an AI Gateway to store AI requests and corresponding results, notice when a new request arrives that is similar or identical to a stored request, and then simply return the stored result rather than re-processing the request. For example, imagine a simple conversational AI that responds to basic questions. A user may ask "Who was the 22nd President of the United States?" An AI system may respond "Grover Cleveland was the 22nd President." Sometime later, another user asks the same question. "Who was the 22nd President of the United States?" Without caching, an AI Gateway would send this request to an AI model, which would (in most cases) give the exact same response as before. In an AI Gateway with caching enabled, the Gateway would recognize that this request has already been processed. Then, instead of sending the request to an AI model, it would look up the stored response and send it directly back, avoiding a call to an AI model altogether. The diagram below illustrates caching:



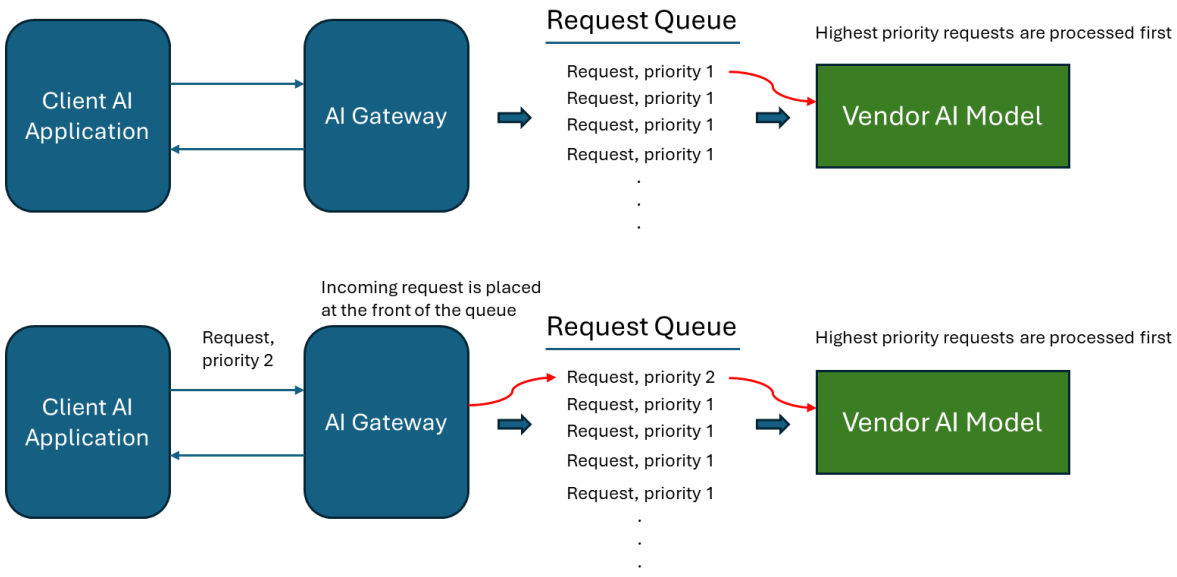
Caching is useful for three important reasons: (1) Caching can reduce latency and speed up overall performance in AI applications. In general, cache lookup and retrieval is significantly faster than AI model processing. Cached response times can be reduced from hundreds of milliseconds to less than 50 milliseconds in some applications. (2) Caching can reduce costs associated with AI model use. Most AI models charge a fee each time they process an AI request. By avoiding extraneous AI model requests, caching can reduce AI usage costs by 20-30% in some applications. (3) Caching can reduce overall load on AI resources. Especially in resource constrained environments, or in cases where AI requests may be throttled by vendors, caching can improve overall throughput and reduce the number/size of AI resources needed in an application.

Author: William Klein, bill.klein@y2ai.co, https://y2ai.co

Message Prioritization:

Not all AI requests are equivalent in their priority or need to be processed as quickly as possible. There are numerous circumstances in which a request may be prioritized ahead of other requests, for example in conversational applications that require ultra-low latency. In other cases, requests may have far lower priority, for example if a request may take a long time to complete (e.g., image or document processing requests) or when the application may not need to be notified of a request completion at all (e.g., batch processing over larger data sets). There are also situations in which priority may not indicate speed of processing, but instead may indicate other factors such as cost requirements or even quality of service needs.

Message prioritization is a mechanism that allows applications to specify the urgency and priority of an individual AI request. An AI Gateway can then optimize both the ordering of requests as well as the use of available AI resources to fulfill requests based on priority information given. An example of message prioritization is shown below:



Author: William Klein, bill.klein@y2ai.co, https://y2ai.co

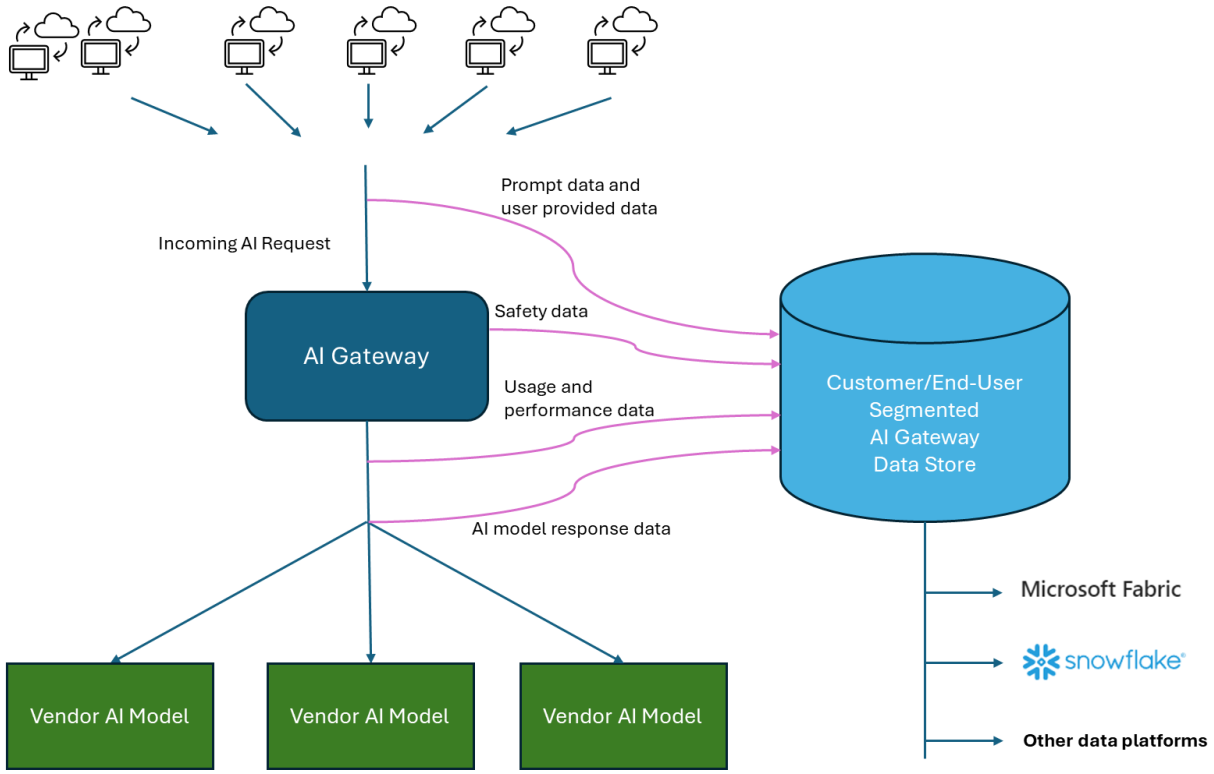
Data Aggregation:

A supremely important function of production AI applications is the ability to track data flowing through the system. As application developers build new user-facing client applications that gather text, voice, and even physical telemetry they are frequently faced with the challenge of recording that data and sending it to a central repository where it can be stored, analyzed, and made useful to an organization. Ad-hoc methods, such as logging data to local files that are later uploaded to a central repository, are extremely common. These methods are also a) hard to maintain, b) error prone, and c) extremely susceptible to data security risks. Even for systems that record data in centralized servers, it can be extremely difficult to capture 100% of the data being transmitted from end-user to application to AI model and back again.

AI Gateways represent an optimal point in the application architecture to capture, store, and aggregate this data. AI Gateways can record every aspect of AI requests and responses, as well as the meta data associated with them. This can include (but is not limited to): AI request prompts, user data and user utterances (text and voice inputs), AI model responses (including text, audio, image, and video), latency, speed, and cost data, and even content safety violations. And because of the positioning of an AI Gateway in the AI pipeline, data collection and aggregation can be performed with high levels of data security, individual user data privacy, and high levels of data and application traceability. The Data Aggregation capabilities of an AI Gateway can be the most critical component in an AI systems architecture, as it can define an organization's compliance and reporting capabilities.

Although AI Gateways can serve as the repository for collected data, a more common practice is to link the AI Gateway to a separate data platform, including systems like Microsoft Fabric and Snowflake. Generally, there are dozens of data storage, security, analytics, and reporting platforms available for AI Gateways to feed data into. The illustration below illustrates how AI Gateways can capture, aggregate, and distribute AI request data.

Author: William Klein, bill.klein@y2ai.co, https://y2ai.co



Summary:

AI Gateways can be a critical component to an organization's AI infrastructure as they scale and grow their use of AI models and resources. An AI Gateway's routing and load-balancing capabilities are crucial to high-availability, high-reliability applications, especially for mission critical or large-scale applications. AI Gateways assist in both simplifying and improving the design and quality of AI systems by removing vendor-specific code, abstracting AI functions, and providing a means to manage and modify AI resource allocations without the need for application redeployment. AI Gateways improve performance, reduce costs, and increase an organization's flexibility and optionality in choosing which AI vendors to work with. And AI Gateways are an optimal choice for managing and aggregating AI-related data safely and securely.

Y2AI:

This whitepaper is presented by Y2AI, an AI tools and infrastructure company that provides a premier AI Gateway service. Visit https://y2ai.co or contact bill.klein@y2ai.co to learn more.

Author: William Klein, bill.klein@y2ai.co, https://y2ai.co