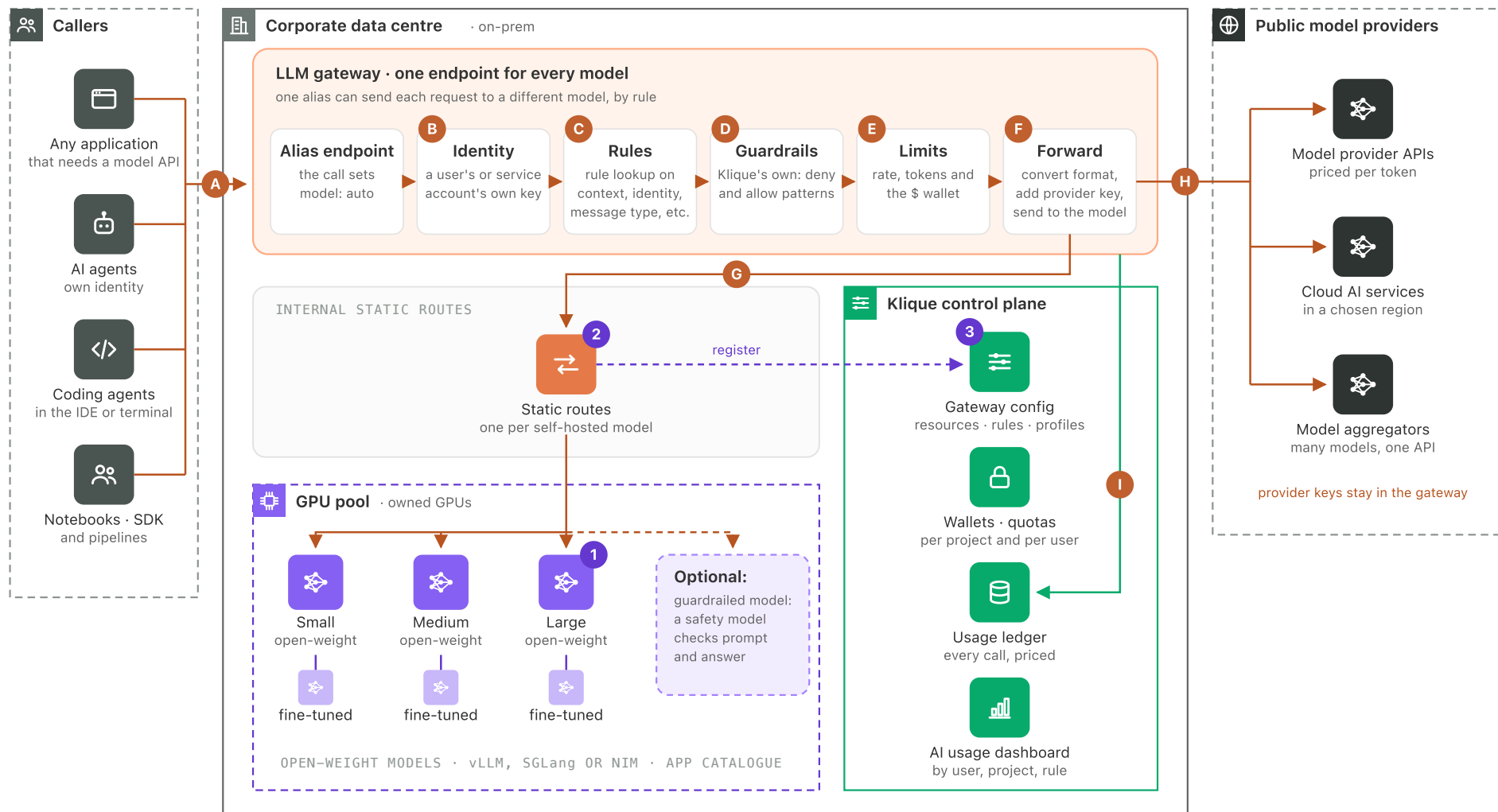


LLM Gateway and Tokenomics Reference Architecture

This reference architecture covers the following scenario: an enterprise IT department provides LLMs as a service to the business. One Klique LLM gateway sits in front of open-weight models on the company's own GPUs and the public model APIs the company contracts with. Callers use model aliases that IT publishes, and one alias can send each request to a different model, chosen by a rule lookup on the request. The gateway checks each call, sends it to the model chosen for it, applies budgets and quotas, and records the cost against a user and a project. The document covers the components, the routing rules, the guardrails, and how spend is limited and attributed.

GATEWAY One, in the corporate data centre, in front of every model	MODELS Open-weight models on owned GPUs, and contracted public model APIs
RUN BY IT, as a service to business units and their projects	AUDIENCE IT architects and sales engineers

How to read the diagram: lettered badges follow one model call, numbered badges follow IT bringing a self-hosted model online.



STEP	DETAIL	LOCATION	CONNECTION
ONE MODEL CALL			
A Call an alias	An application, AI agent, coding agent etc. sends a request to its gateway address with its own key, a user's or a service account's. The model field holds an alias, such as auto, not a provider's model name.	Corporate network	Caller → gateway, HTTPS
B Identity	The gateway validates the key and takes the user or service account from it. It checks that the profile gives that caller the alias. A project or task named in the request headers is checked against what the caller may access.	LLM gateway	Gateway → control plane
C Rules	The alias points at one AI resource or at a routed resource. For one resource, the call goes to that model, balanced across its endpoints by weight. For a routed resource, a rule lookup on the request's context, the caller's identity, the message type and any other field of the request picks the model.	LLM gateway	None: a routing decision
D Guardrails	Klique's own guardrails, built into the gateway: deny and allow patterns run over the prompt. The chosen model's own patterns apply if it has them, otherwise the global ones. A deny match is refused before any model is called.	LLM gateway	None: checked in the gateway
E Limits	The model's rate limit, the caller's token quota, and the project and user wallets are checked. When a token quota or wallet is reached, the rule passes the call to the next rule by default, or refuses it if set to. At a rate limit it rejects the call by default, or passes it on if set to.	LLM gateway	Gateway → control plane
F Forward	The gateway converts the request to the model provider's API format if it differs, adds the provider key it holds, and sends the call to the chosen model: one of the self-hosted models on owned GPUs, or a model at a public model provider.	LLM gateway	None: in the gateway
G Self-hosted models	Self-hosted models are reached through their internal static routes: open-weight models, and optionally a guardrailed model that checks the prompt and the answer.	GPU pool	Stays in the data centre
H Public model providers	Models at public model providers are called over the internet with the provider keys held in the gateway.	Internet	Gateway → provider, HTTPS
I Meter and price	The call is recorded on the usage ledger with its user, project, profile, alias, rule, token counts and latency, priced at the model's rate.	Control plane	Gateway → control plane

STEP	DETAIL	LOCATION	CONNECTION
IT BRINGS A SELF-HOSTED MODEL ONLINE			
1 Deploy	IT launches an open-weight model from the app catalogue on owned GPUs, with vLLM, SGLang or NVIDIA NIM.	GPU pool	Klique worker daemon starts it
2 Publish	The model is published on an internal static route, an address that stays the same across restarts.	Internal route	None: configuration
3 Register	IT registers the static route as an AI resource, then points an alias or a rule at it. Callers use the alias; their code does not change.	Control plane	None: configuration

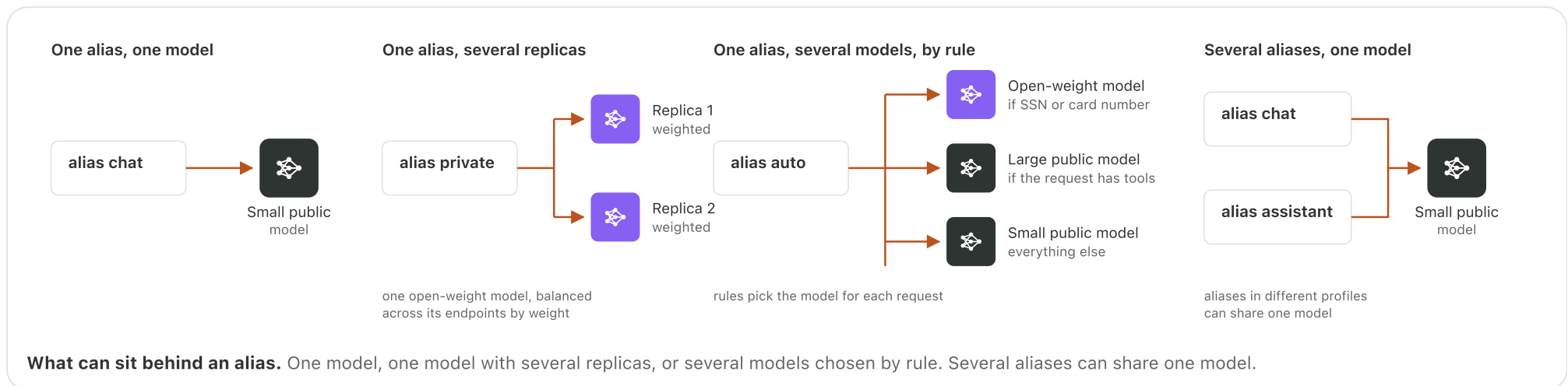
Provider keys stay in the gateway. A caller's key identifies the caller and reaches only the aliases in the caller's profile.

Business value

- **For IT: one endpoint for all models.** Open-weight models on owned GPUs and public model APIs are reached through one gateway and one set of aliases. One alias can send each request to a different model, chosen by a rule lookup on the request's context, the caller's identity and the message type. IT can change these models and rules, add a provider or move a workload to owned GPUs without changes to client code.
- **For finance: spend limits and cost attribution.** Wallets set dollar limits per project and per user and are checked before each call. Every call is priced and recorded against a user and a project, which gives the data for chargeback.
- **For security and risk: control over where prompts go.** Rules send prompts that match sensitive patterns, or come from named projects, to models on owned GPUs. Guardrails run in the gateway, so every client is subject to them.

Alias endpoints and keys

Callers point a standard model API client at the gateway endpoint for their **profile** and set the `model` field to an **alias**, such as `auto` or `chat`. Aliases belong to profiles. Each profile has its own endpoint URL and is open to the users, groups or projects named on it, or to everyone if none are named. An alias points at one AI resource, which can have several weighted endpoints, or at a routed resource, whose rules choose between several models. Several aliases can point at the same model. Models are reached only through an alias, never by their own name.



IT decides what each alias points at and can change it at any time, for example to a new model version, another provider, or a model on owned GPUs. Callers keep the same alias, so client code does not change. A profile can also name a default model, used when a request names no model.

Clients can send requests in the OpenAI Chat, OpenAI Responses, Anthropic Messages or Bedrock Converse API format. The gateway converts each request to the format of the model's provider where they differ. Coding agents can use the gateway directly.

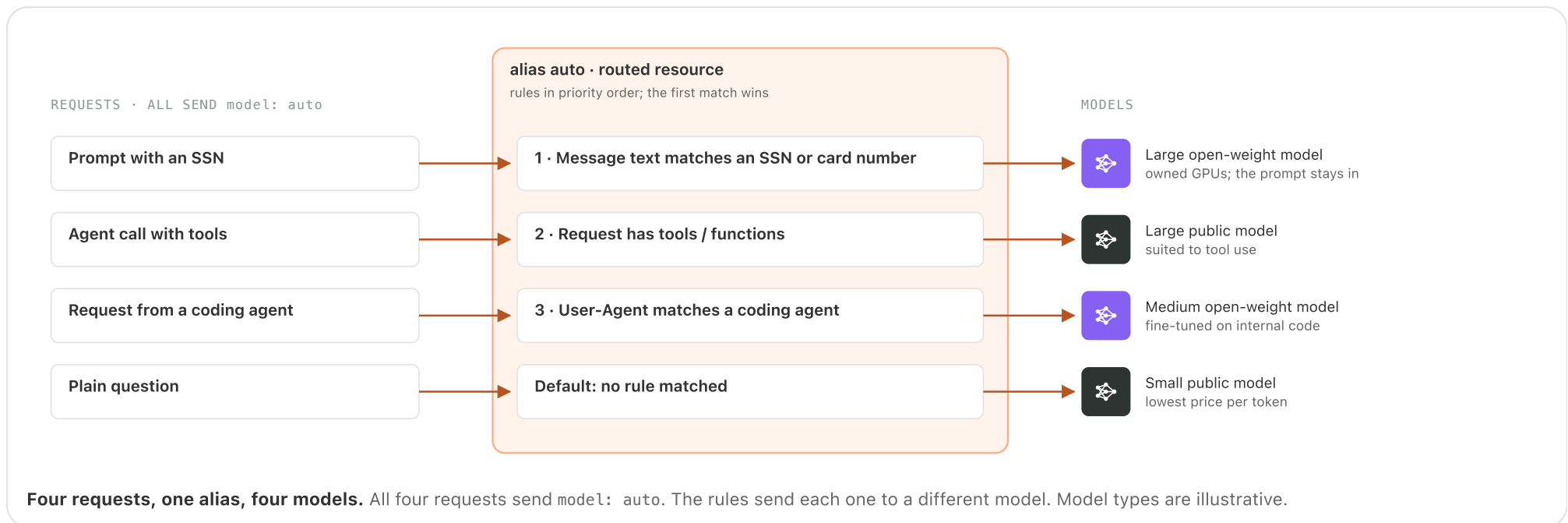
A caller's key is an identity, not a provider credential. Users create and revoke their own gateway keys, with or without an expiry date. The key is the user's own Klique token, sent as a Bearer token or in an Anthropic `x-api-key` header, so Claude Code works with the gateway as is. It identifies the caller and holds no provider credential. It reaches only the aliases in the user's profile, within the

user's wallet and quotas, and the user or IT can revoke it. An application uses a service account with its own key, so its usage is recorded separately. Provider keys are held in the gateway and rotated there.

THE CALLER SENDS	WHAT THE GATEWAY DOES
Its key, as a Bearer token or in an Anthropic x-api-key header	Validates it and takes the user or service account from it. A caller cannot set its own identity.
An alias in the model field	Looks it up in the profile and resolves it to a resource or a routed resource. An alias the caller has no access to is refused.
A project ID header	Checks that the caller may access the project. Rules and usage records can rely on the checked project ID.
A task ID header	Records the call against that task and the task's project, so the model spend of a training or evaluation run is recorded with the run.
The User-Agent and tier headers, and the user field in the request body	Makes them available to rules. The client sets these values and the gateway does not verify them, so they are not suitable for rules about cost or compliance.

One alias, several models

An alias that points at a routed resource sends each request to a model chosen by rules. Callers use one model name, such as `auto`. For each request, the gateway chooses the model by a rule lookup on the request's context, the caller's identity, the message type and any other field of the request. Callers do not need to know which models exist or which one suits a request.



This allows IT to:

Use a suitable model for each request

Requests with tools go to a larger model, plain questions to a smaller model with a lower price per token, coding agents to a model fine-tuned on internal code, and long prompts to a long-context model.

Keep sensitive prompts on owned GPUs

A prompt that matches a sensitive pattern, or comes from a named project, goes to a model on owned GPUs. The decision is made for each request, not for each application.

Change routing in one place

Rules are edited in the gateway and apply to every caller of the alias at once. Client code does not change.

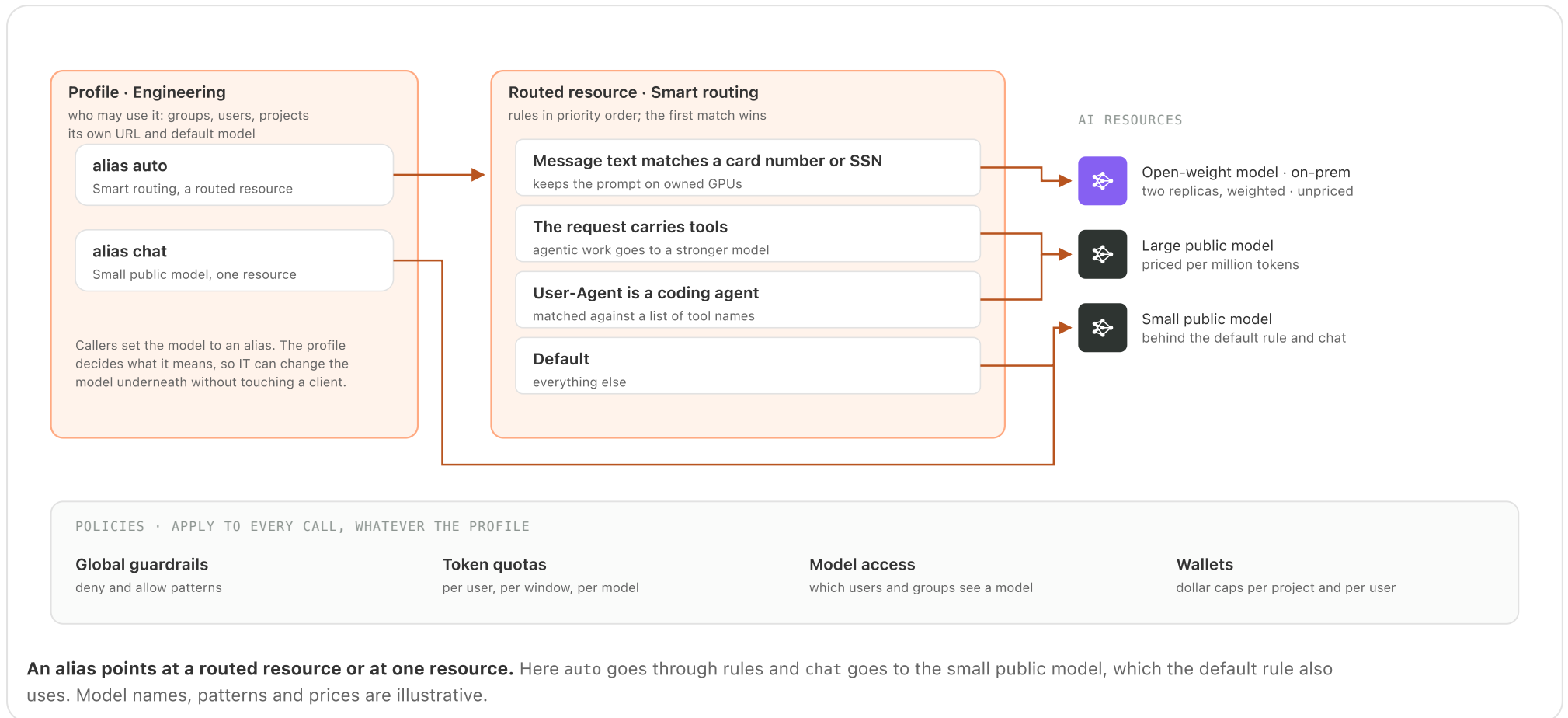
Continue when a rate limit is reached or a wallet is spent

When a model's rate limit is reached or a wallet is spent, a rule can pass the call to the next rule, for example to a self-hosted model.

Each call is recorded with the rule that chose its model, so the usage and cost of each rule can be reported.

Building blocks

The gateway is configured with four kinds of object, in the LLM gateway settings of the Klique UI or through the API. **AI resources** are the models. **Routed resources** hold rules that choose between models. **Profiles** hold the aliases and decide who can use them. **Policies** apply to every call.



OBJECT	HOLDS	EXAMPLE
AI resource	One model at one provider, or one static route to a model on owned GPUs. One or more endpoints with weights, for balancing across replicas. Prices per million tokens for input, output, cache reads and cache writes. A rate limit and guardrails. A tag for grouping usage.	A model from a public model provider; an open-weight model on owned GPUs; a model in a cloud AI service in a chosen region
Routed resource	Rules in priority order, each with a condition, a target resource, and what to do when the target is rate limited or the wallet is spent. A default resource for calls that match no rule.	auto: pick a model per request
Profile	Aliases and what each points at, a default model, its own endpoint, and the users, groups and projects allowed to use it. With no list, everyone may use it.	One profile per business unit, or a profile for one group
Policies	Global guardrails, token quotas, model access by user and group, and wallets per project and per user.	Company-wide limits and blocked patterns

The **Simple Setup** wizard creates a profile, a routed resource served as auto, and the resources it routes to. It asks for the provider keys it needs and reuses resources that already exist. The objects it creates can be edited afterwards like any others.

Routing rules

A routed resource checks its rules in priority order. The highest-priority rule whose condition matches the request decides the model. If no rule matches, the default resource serves the call. Each rule tests one field of the request with an operator: equals, a regular expression, or greater or less than a number. Rules test the request as the client sent it, before any format conversion.

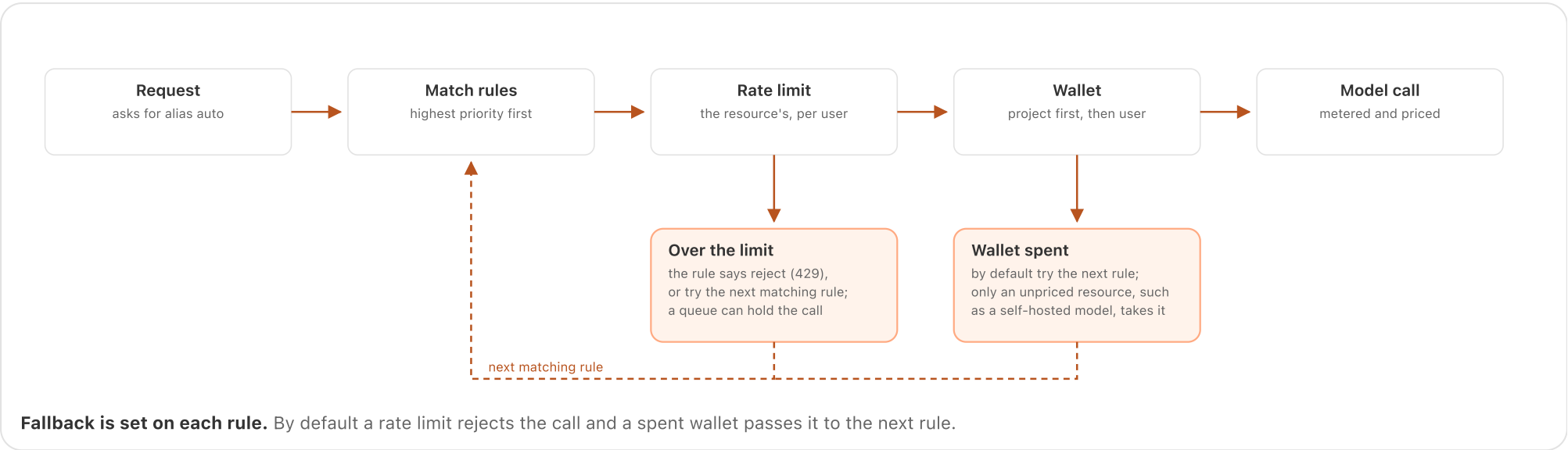
A rule can test any field of the request: its content, the caller's identity and project, the client, the generation settings, the request path and method, and any custom body field, header or query parameter. Rules can be combined in any order of priority, so routing can be as simple or as detailed as needed.

GROUP	FIELDS A RULE CAN TEST	USED FOR, FOR EXAMPLE
Content	Message text (regular expression), has tools / functions, has image, prompt size in estimated tokens	Sensitive data to owned GPUs; tool use, images and long prompts to suitable models
Identity and context	Klique project ID, client / SDK (User-Agent), client user tag (the user field in the body), entitlement tier header	A project's requests to a set model; coding agents to a fine-tuned model
Model	Requested model name	A caller that names a model within a routed alias
Generation settings	Max output tokens, temperature, top-p, seed, streaming	Large outputs or streaming requests to a model suited to them
Request	Request path, HTTP method, request size in bytes	One API path, or very large requests, to a set model
Custom	Any body field, header or query parameter	Fields an application adds, such as a workload type or a department

The Klique project ID is checked by the gateway before rules run, and the user comes from the key. The User-Agent, the client user tag, the entitlement tier header and custom fields are set by the client and not verified: they suit rules about model choice, not rules about cost or compliance. Message text and prompt size have limits: a regular expression matches text, not meaning, and prompt size is an estimate of characters divided by four. Images are not converted between API formats yet, so requests with images go to a model that accepts the client's format.

Rate limits and spent wallets

Each rule sets what happens when its model cannot take the call. When the model's rate limit is reached, the rule either rejects the call or passes it to the next matching rule; the resource can instead hold calls in a short queue. When the wallet is spent, the rule passes the call to the next rule by default. Only an unpriced resource, such as a self-hosted model, can serve a call once the wallet is spent.



Suggested rules

The rules below cover common requirements. Each is one rule or a short chain of rules in a routed resource, and several can be combined under one alias such as auto. In each line the first item is the condition and the next items are where the call goes.

Rules for sensitive data

Card numbers and SSNs to an open-weight model

Message text iregex card number or SSN →

Open-weight model · owned GPUs

A Message text rule with patterns for card numbers and social security numbers sends matching prompts to a model on owned GPUs. Other prompts go to the next rule. Internal project names or customer identifiers can be added to the same pattern. Give this rule the highest priority so no cost rule is checked first.

```
\b(?:\d[ -?]{13,16}\b
\b\d{3}-\d{2}-\d{4}\b
```

A regulated project to an open-weight model

Klique project ID = Claims →

Open-weight model · owned GPUs

All requests from one project go to a model on owned GPUs, whatever the client. The gateway checks the project ID, so a client cannot change it with a header.

A project to the NIM Guardrails app

Klique project ID = HR → NIM Guardrails app →

Safety model checks prompt and answer

Requests that need a check on meaning rather than on a pattern go to the NIM Guardrails app, where a safety model checks the prompt and the answer. See guardrails.

Block a pattern on every call

Global deny pattern → **Refused**

For prompts that must not reach any model, such as prompts containing credentials, use a deny pattern in the global guardrails instead of a rule. It applies to every model that has no patterns of its own, in every profile.

Rules for cost and model choice

Requests with tools to a larger model

Has tools = true → **Large public model** →

default: small public model

Requests that carry tools go to the larger model and other requests to the smaller, cheaper one. Requests without tools are billed at the lower rate.

Coding agents to a fine-tuned model

User-Agent iregex coding agents →

Open-weight model, fine-tuned on internal code

Coding agents identify themselves in the User-Agent header, and the rule matches a list of their names. The header is set by the client, so use this rule for model choice and use a wallet for cost control.

Long prompts to a long-context model

Prompt size above a threshold → **Long-context model**

Prompts above the threshold go to a model with a larger context window; other prompts go to the default. The size is an estimate, so set the threshold below the model's limit.

Images to a vision model

Has image = true → Vision model, same API format

Requests with images go to a model that accepts the client's image format, because images are not converted between API formats yet.

Rules for limits and fallback

Overflow to a second model at the rate limit

Small public model · per-user rate limit →

next rule: large public model

The first model has a per-user rate limit. When a user exceeds it, the rule passes the call to the next rule instead of returning an error.

Fallback to an unpriced model when a wallet is spent

Large public model · wallet spent →

next rule: open-weight model · owned GPUs

When the project or user wallet is spent, the call passes to the next rule. The self-hosted model has no price, so the wallet does not apply to it. Calls continue on that model until the wallet period resets.

One model for one group

Profile Research · alias auto → Top-tier public model

A separate profile, open to one group, points its alias at one model. The group uses the same alias name as everyone else; the profile and its wallet set the model and the spend.

Simple Setup presets

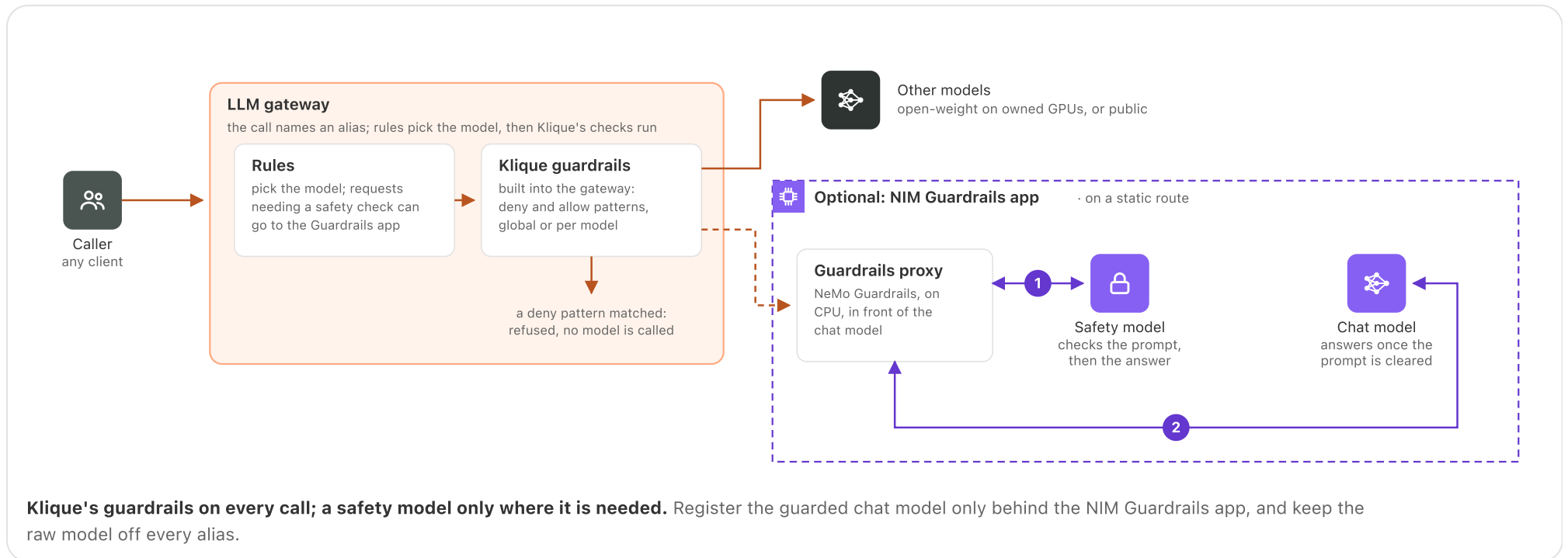
Simple Setup includes several of these as presets, using small, large and top-tier models from one public model provider. Each preset creates a profile serving auto, a routed resource with a catch-all rule below the condition so every request is served, and the resources it routes to. In the rate limit fallback preset, when a resource's rate limit is reached, the call moves to the next rule's model.

PRESET	RULE	ALL OTHER REQUESTS
Rate limit fallback	When a resource's rate limit is reached, the call moves to the next rule's model	The catch-all rule's model
Cost first	Requests with tools to the large model	Small model
Quality first	All requests to the top-tier model	Top-tier model
Long context	Prompts over about 150,000 tokens to the large model	Small model
Vision	Requests with an image to the large model	Small model
Coding	Coding agents, by User-Agent, to the large model	Small model

Content and project rules, such as the sensitive-data rule above, are added in the rule dialog.

Guardrails in two layers

Guardrails run in two layers. Klique's own guardrails are built into the gateway and apply to every call. Optionally, for requests that need a model-based check, the NIM Guardrails app places a safety model before and after the chat model. Most deployments need only the first layer.



Klique's own guardrails, built into the gateway

Every endpoint requires a valid token, and model access limits each model to named users and groups. Deny and allow patterns then run over every message in the prompt, after the rules have chosen the model. A deny match is refused before any model is called. If allow patterns are set, a prompt must match one of them. An AI resource can have its own patterns, which replace the global ones for that resource. Patterns are fast but only match text, not meaning.

Optional: the NIM Guardrails app

The NIM Guardrails app, launched from the app catalogue, runs a NeMo Guardrails proxy on CPU in front of a chat model. A safety model, such as an NVIDIA content-safety model, checks the prompt and then the answer. An optional judge model runs self-check rails. The app is published on a static route and registered as a resource, so an alias or a rule can send calls to it.

Cost of a guardrailed call

Each guardrailed call makes two extra safety-model calls, one for the prompt and one for the answer, which adds latency and GPU time. Use a rule to send only the requests that need it.

Screening and evaluation

Guardrails check a call while it runs. Evaluating the quality of answers afterwards is a separate task and is not covered here.

Budgets, quotas and limits

Several controls limit usage and spend. They are checked before the model is called, and each has a set result when it is reached.

CONTROL	SET ON	COUNTS	WHEN IT IS REACHED
Wallet	Policies: per project and per user, with defaults and overrides	Dollars per day, week or month, from each resource's prices	The project wallet is checked first, then the user's. By default the rule passes the call to the next rule, where only an unpriced resource can serve it; otherwise the call is refused
Token quota	Policies: per user, with overrides per model, group and user	Tokens per window: total, prompt only or completion only	By default the rule passes the call to the next rule; a rule set to reject refuses it
Rate limit	Each AI resource, with group and user overrides	Requests per window, per user	Rejected with a 429 by default, or held in a short queue; a rule can pass the call to the next rule
Model access	Policies: per model	Which users and groups may call it	Refused
Profile access	Each profile	Which users, groups and projects may use it	Refused

Set a price on every paid model

Wallets count dollars from each resource's prices per million tokens, including cache reads and writes. A public model with no price set does not count against any wallet.

Keep one resource unpriced

A self-hosted model with no price is not limited by wallets. A rule chain that ends on one continues to serve calls after a wallet is spent. To charge back its cost, use GPU usage reports rather than a price in the gateway.

Users can see their own spend

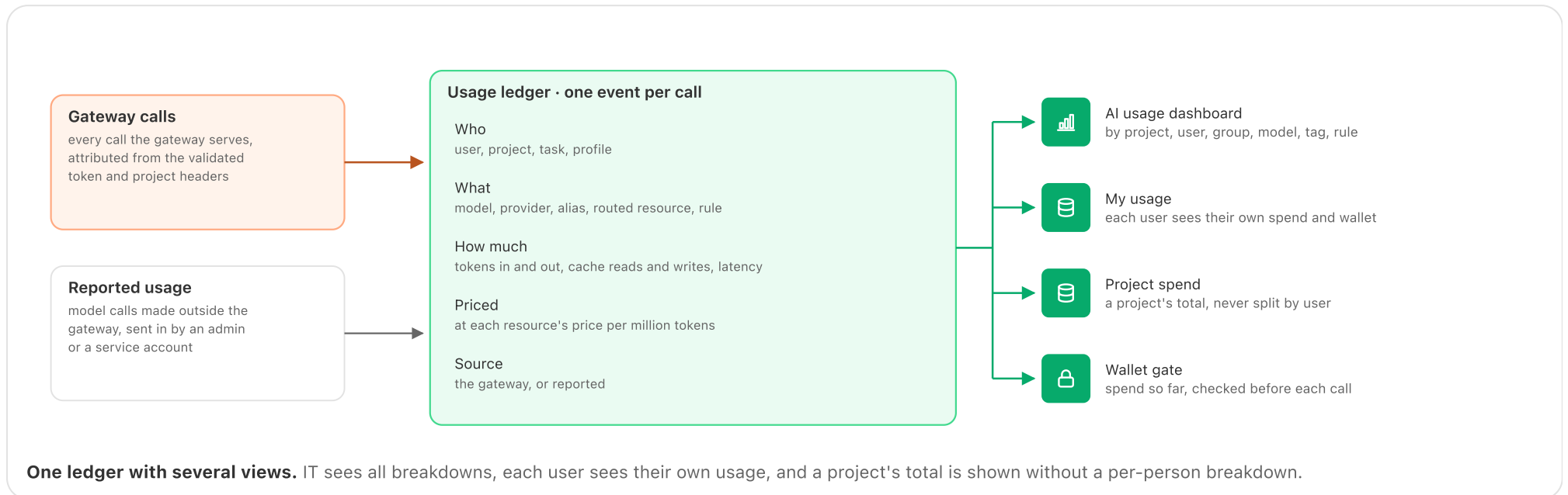
Any user can see their own usage and wallet balance for the day, week or month, in dollars.

The wallet check fails open

If the wallet check cannot complete, the call goes ahead and is still recorded on the ledger.

Metering and attribution

Every call is recorded as one usage event on the ledger, with its token counts and latency, priced at the rate of the resource that served it. Usage outside the gateway can be added to the same ledger by an admin or a service account, so spend on a provider's own console can be reported alongside gateway spend.



The user is taken from the key and the project from a header the gateway has checked, so the client cannot set either. Cost can be broken down by the following, alone or combined:

User and group

The user from the validated key, and the user's groups. Used for chargeback by team.

Project and task

The checked project, or the project of the task that made the call. The model spend of a training or evaluation run is recorded with the run.

Profile and alias

The profile and the alias the call used, which show usage by business unit and by service.

Routed resource and rule

The rule that chose the model. Shows the cost of each rule and how often a rule, such as the sensitive-data rule, is used.

Model and provider

The spend per provider, and the share of calls served on owned GPUs.

Resource tag

A free-text tag on each resource, such as a cost centre or a contract, for grouping spend the way finance reports it.

Self-hosted open-weight models

Open-weight models on owned GPUs keep prompts in the data centre, have no per-token charge, and can serve calls after a wallet is spent. Klique runs them on the same platform as the gateway. The serving apps are scheduled like any other work, on whichever GPU pool their compute resource lists: Kubernetes, an HPC cluster, bare metal, a public cloud provider or a neocloud. The [Meta-Scheduler Reference Architecture](#) covers how.



Deploy

Launch vLLM, SGLang or NVIDIA NIM from the app catalogue on a compute resource backed by owned GPUs. A Klique worker daemon starts it like any other workload, under the same quotas and priorities.

Publish

Give the deployment an internal static route. The address stays the same across restarts and redeployments.

Register

Add the static route as an AI resource with the provider type for self-hosted model servers. Each replica is an endpoint with a weight, and the gateway balances calls across them.

Point an alias or rule at it

Use the resource as the target of an alias, a rule or a fallback. Callers use an existing alias and see no change in their code.

Running LLMs as a service

With these components, IT can run model access as a shared service. Klique provides GPU scheduling, model serving, the gateway and metering in one platform. The agents that call the gateway get their compute from the same scheduler, so GPU-hours and tokens are governed with one identity and one set of policies; the [Meta-Scheduler Reference Architecture](#) covers the compute side.

WHO	DOES	WHERE
IT platform team	Registers resources and prices, writes rules, sets guardrails and policies, creates a profile per business unit	LLM gateway settings, app catalogue
Finance	Sets wallets per project and per user, reviews spend by project, tag and provider, runs chargeback	AI usage dashboard
Business unit or project lead	Requests a profile, holds a project wallet, sees the project's total spend	Project spend
Developer or user	Creates their own key, calls an alias from any client, sees their own spend and wallet	Their own settings
Application owner	Gets a service account and key for the application, which calls an alias like any other client	A service account

Publish a list of aliases

Publish a short list of aliases, such as auto, chat, code and private, with the purpose of each. The model behind an alias can change without changes to clients.

Grant access by group

Adding a group to a profile gives its members the profile's aliases, with the profile's wallets and quotas. Removing a user from the group removes the access.

Set prices for chargeback

Resource prices are used for both wallets and chargeback. They can be set at cost or with a margin.

Design considerations

Use verified fields for policy rules

The user comes from the key and the project ID is checked by the gateway, so rules on them can be used for cost and compliance. User-Agent, the tier header and the client user tag are set by the client.

Patterns match text, not meaning

Message text rules and deny patterns are regular expressions. They match formats, such as card numbers, not meaning. For checks on meaning, use the NIM Guardrails app.

Put sensitive-data rules first

If a cost rule has a higher priority than a sensitive-data rule, a prompt can be sent to a public model before the sensitive-data rule is checked. Give sensitive-data rules the highest priority.

Prompt size is an estimate

It is characters divided by four. Set long-context thresholds below the target model's limit.

Images are not converted

Images are not converted between API formats yet. Send requests with images to a model that accepts the client's format.

Keep an unpriced fallback

An unpriced self-hosted model at the end of a rule chain serves calls when a rate limit is reached or a wallet is spent.

Rotate provider keys in the gateway

Provider keys are stored only in the gateway. Rotating one is a single change and does not affect callers' keys.

Guardrailed calls add latency

Two safety-model calls per request add latency and GPU time. Use a rule to send only the requests that need them.

Keep the ledger as the audit record

The metering ledger records every call with the user from the key, the checked project, its profile, alias, routed resource, rule, model and provider, token counts including cache, and latency. Each call has its own record, for audit and chargeback.

When something fails

At a rate limit, the call is rejected with a 429, held in a short queue or passed to the next rule, by setting. At a token quota or wallet limit, it passes to the next rule or is refused. The wallet check fails open. Unpriced resources are never limited by wallets, so a self-hosted fallback keeps serving after a wallet is spent.

One gateway per tenant

In a multi-tenant deployment, each tenant has its own LLM gateway, configured by the tenant admin. The [Multi-Tenant reference architecture](#) covers tenants.

Glossary

LLM gateway	The endpoint for all model calls. It handles identity, profiles and aliases, routing rules, Klique's own guardrails, limits, format conversion and metering.
Alias	The model name a caller sends, such as auto. It belongs to a profile and points at one AI resource or a routed resource. Several aliases can point at the same model.
Profile	A set of aliases with its own endpoint, a default model, and the users, groups and projects allowed to use it.
AI resource	One model at one provider, or one static route to a self-hosted model, with its endpoints, prices, rate limit and guardrails.
Routed resource	Rules in priority order that choose an AI resource for each request, with a default for requests that match no rule.
Rule	A condition on one field of the request, a target resource, and what to do when the target is rate limited or the wallet is spent.
Wallet	A dollar limit per project or per user for a day, week or month, checked before each call.
Token quota	A token limit per user per window, with overrides per model, group and user.
NIM Guardrails app	A NeMo Guardrails proxy, on CPU, in front of a chat model, where a safety model checks the prompt and the answer.
Static route	A fixed internal route to a self-hosted workload, such as a model endpoint.
Usage ledger	One priced record per model call, with its user, project, task, profile, alias, rule, model, token counts and latency.

Simple Setup

The wizard that creates a profile, a routed resource and its resources from a preset.

Klique AI · LLM gateway and tokenomics reference architecture · one gateway in front of open-weight and public models. Model names, patterns, limits and prices are illustrative.