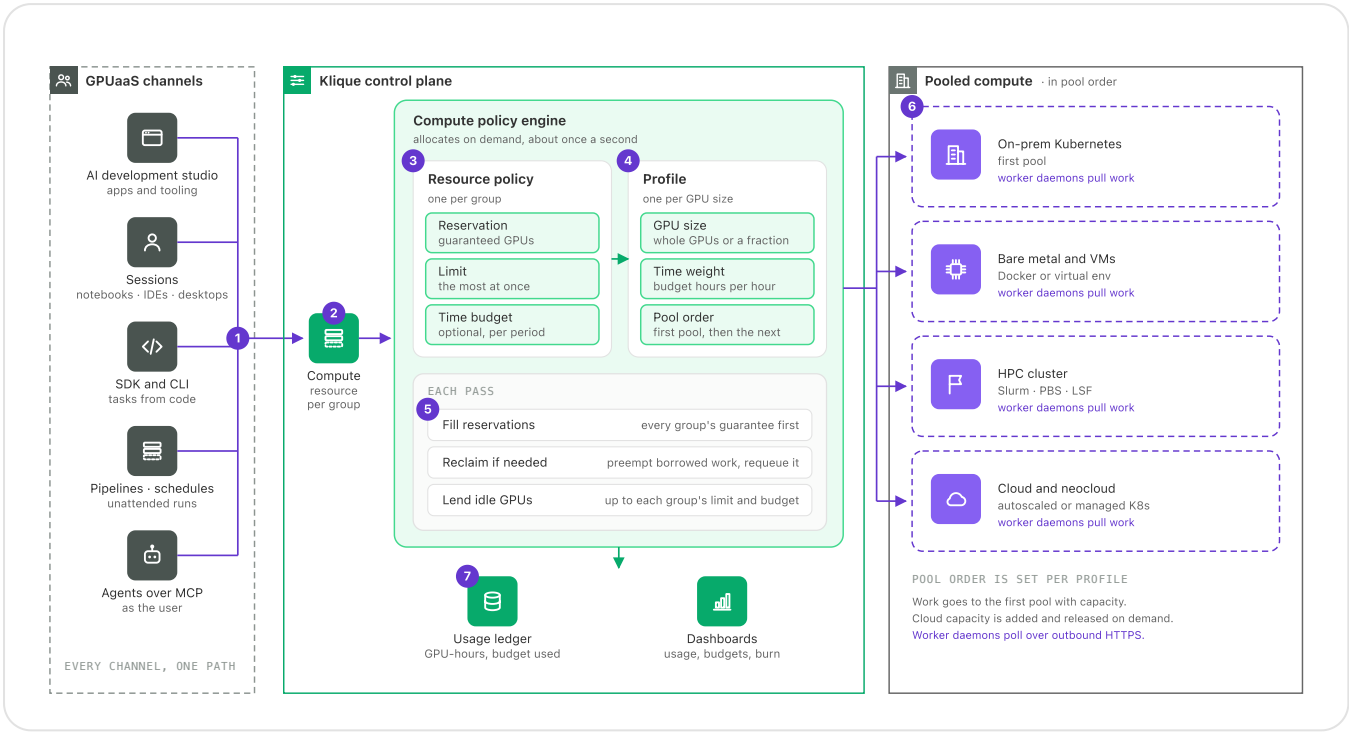
Resource Allocation and Cost Governance Reference Architecture

This reference architecture covers the following scenario: one organisation shares a GPU estate across many groups. The estate spans on-prem Kubernetes, bare metal and VMs, HPC clusters, public clouds and neoclouds. Users and agents request GPUs through GPUaaS channels. IT gives each group a reservation and a limit, and can add a time budget per day, week or month. The Klique compute policy engine allocates GPUs on demand against those rules, in the pool order IT sets, and records every GPU-hour against the user, project and group. The document covers the allocation model, how idle GPUs are lent and reclaimed, how time budgets and weights work, and what each stakeholder controls and sees.

ESTATE On-prem Kubernetes, bare metal and VMs, HPC, cloud and neocloud	SHARED BY Groups: teams, research groups, business units, projects
RUN BY IT or research computing, as a service	AUDIENCE IT architects and sales engineers

How to read the diagram: numbered badges follow one compute provisioning task from request to GPU.



STEP	DETAIL	LOCATION	CONNECTION
1 **Request**	A user or an agent asks for GPUs through a GPUaaS channel: an app or tool in the AI development studio, a session, the SDK or CLI, a pipeline or schedule, or an agent over MCP. Every channel uses the requester's own identity.	Access channel	Client → control plane, HTTPS
2 **Compute resource**	The request lands on a compute resource. Each one joins a group's resource policy to a profile, so it says who is asking and for what size of GPU. It carries no location.	Control plane	None: a record
3 **Policy**	The compute policy engine checks the group's policy: its reservation, its limit and, if set, the time left in its budget.	Compute policy engine	None: a check
4 **Profile**	The profile gives the task's GPU size, whole GPUs or a fraction, its time weight and the ordered list of pools it may use.	Compute policy engine	None: a check
5 **Allocate**	On its next pass, about once a second, the engine fills every group's reservation first, reclaims borrowed GPUs if a reservation is missed, then lends idle GPUs up to each group's limit and budget.	Compute policy engine	None: a decision
6 **Place**	The task goes to the first pool in the profile's order with capacity. A Klique worker daemon in that pool pulls it and starts it. Cloud pools add GPUs on demand and release them when idle.	Pool	Worker daemon → control plane, outbound HTTPS
7 **Record**	GPU time is recorded on the usage ledger against the user, project and group, and charged to the group's time budget at the profile's weight.	Control plane	Worker daemon → control plane

Rules live in the control plane and apply to every channel and every pool. Adding a pool or a channel changes no rule.

Business value

- **For the CIO: a GPU estate run as one service.** Every GPU, wherever it runs, is shared under rules the business sets. Idle GPUs are lent to groups with work instead of sitting reserved, which raises use of the hardware already paid for. Cloud capacity is added when owned capacity is full and released when it is idle.
- **For IT and research computing: allocation as policy.** Each group's share is a reservation, a limit and an optional time budget, set once and applied to every channel and pool. Groups self-serve; IT changes a rule rather than handling a ticket.
- **For finance: spend that can be planned.** Time budgets turn GPU access into hours per period, with premium GPUs weighted to draw budgets down faster. Every GPU-hour is recorded against a user, project and group for showback and chargeback.
- **For developers and researchers: GPUs on demand.** They request a GPU size and get the first suitable capacity. Work that is preempted goes back to the front of its queue and runs again, so idle GPUs can be borrowed without losing work.

01

Who decides what

Allocation touches every level of the organisation. Each stakeholder sets or sees a different part of it, and all of it lives in one control plane.

STAKEHOLDER	DECIDES	SEES
CIO and IT leadership	How the estate is shared between business units, and how much cloud spend is acceptable	Use of the estate, GPU-hours and spend by business unit, cloud capacity in use
IT director, platform team or HPC centre	Pools and profiles, the time weight of each GPU type, pool order, and each group's reservation, limit and time budget	Capacity and use per pool, queues per compute resource, budgets near their limit
Department or business-unit head	How the department's GPUs and hours per period are shared between its teams and projects	GPUs and hours used per team, work waiting, budget left in the period
Finance	Prices per GPU type, how spend is attributed	GPU-hours and cost by group, project and user, for showback and chargeback
Team lead or principal investigator	Who in the group uses the group's share, which projects get priority	The group's use against its reservation, limit and budget, and when the budget will run out
Developer or researcher	Which GPU size to request, from which channel	Where the work is: waiting, running, or requeued and why

02

The allocation model

Three objects describe the whole estate. **Pools** are the hardware, each with a declared capacity: a Kubernetes cluster, a set of bare-metal servers, an HPC partition, an autoscaled cloud account. **Profiles** are GPU sizes: whole GPUs or a fraction, with a time weight and an ordered list of pools. **Resource policies** belong to groups: a reservation, a limit and an optional time budget. A **compute resource** joins one policy to one profile, and is what users and agents send work to.

Reservation

The GPUs a group is guaranteed. The engine fills every reservation before it lends any idle capacity. Reservations across groups should fit the capacity of the pools behind them.

Limit

The most a group can use at once, including borrowed GPUs. A profile larger than a group's limit cannot be connected to that group's policy.

Time budget

Optional. The budget hours a group may spend per day, week or month, on top of its reservation and limit.

Execution priority

Within one pool, which profile's work is placed first, for example small tasks before large ones. Set by the order of profiles on the pool.

Queuing priority

Within one profile, which group's work is placed first when several groups are waiting. Set by the order of policies on the profile.

Pool order

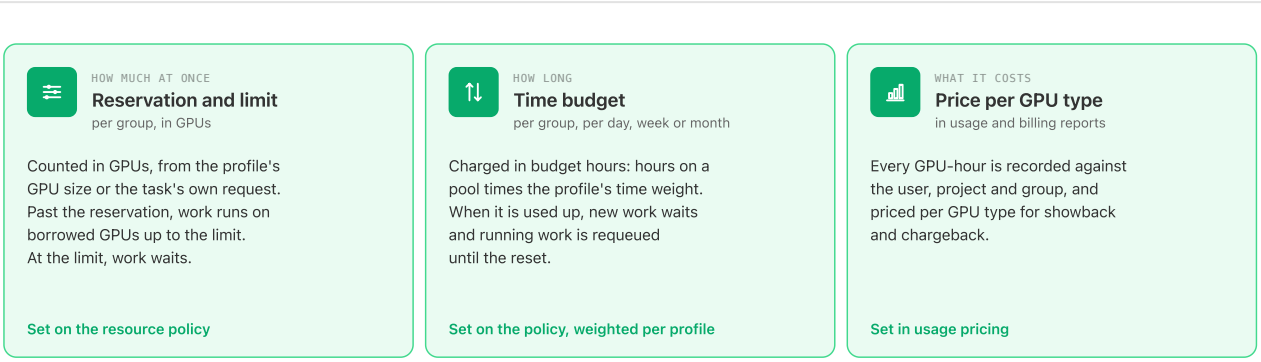
Each profile lists the pools it may use, in order. The engine places work in the first pool with capacity, then the next.

Policies sit above the pools, so a group's reservation holds in whichever environment runs its work, and adding a pool changes no policy. The Meta-Scheduler Reference Architecture covers pools across environments in depth; this document covers how the rules on top of them allocate and govern GPUs.

03

Three controls

GPU allocation answers three separate questions, and each has its own control. Keeping them apart lets IT set a generous limit for use of idle GPUs without losing control of the total over a period.



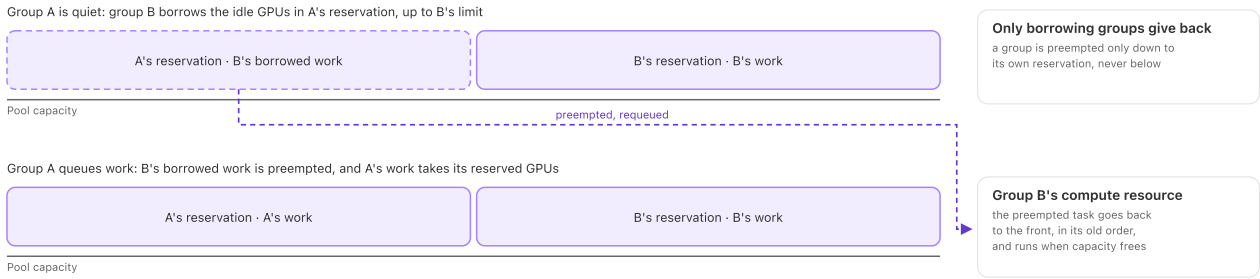
Three controls. Reservation and limit govern GPUs in use at once, the time budget governs hours over a period, and pricing turns recorded GPU-hours into cost.

CONTROL	UNIT	WHEN IT IS REACHED	SET BY
Reservation	GPUs, from the profile's size or the task's own request	Further work runs on borrowed GPUs, up to the limit	IT, per group
Limit	GPUs	Further work waits in its compute resource	IT, per group
Time budget	Budget hours: hours on a pool × the profile's time weight	New work waits and running work is requeued until the period resets or the budget is raised	IT, per group, with weights per profile
Price per GPU type	Currency per GPU-hour	Reporting only: prices never stop work	Finance and IT

04

Lending and reclaiming idle GPUs

A reservation is a guarantee, not a fence. When a group is not using its reserved GPUs, other groups can borrow them, up to their own limits. When the owning group queues work, the engine takes the GPUs back.



Borrowing and reclaim. Idle reserved GPUs are lent; when the owner needs them, borrowed work is preempted and requeued. Groups are illustrative.

Each pass of the compute policy engine runs in the same order:

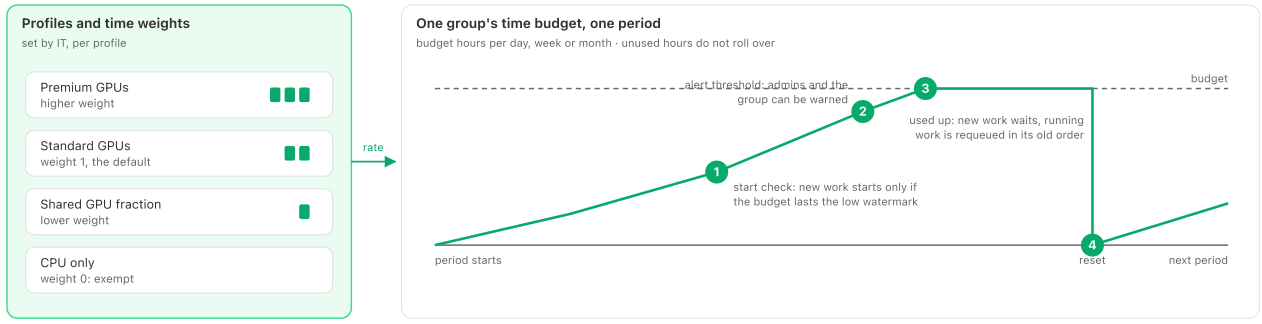
PASS	WHAT THE ENGINE DOES
1 **Fill reservations**	Places waiting work for every group that is below its reservation, first pool first.
2 **Reclaim**	If a group below its reservation still has work waiting, preempts work from groups above their reservation, only as much as the waiting work needs and never below a group's own reservation. Preempted work goes back to the front of its compute resource, in its old order.
3 **Lend**	Places remaining work on idle GPUs, up to each group's limit and within its time budget, first pool first.

Preempted work is not lost: it is requeued and runs again when capacity frees. Training code that checkpoints resumes from its last checkpoint; work that does not checkpoint starts again. "Full" means a pool's declared capacity is used, so declared capacity should match the hardware.

05

Time budgets and weights

A reservation and a limit control how many GPUs a group holds at once. They do not limit how long. A group that stays inside its limit can hold those GPUs for a whole month. A time budget adds that second dimension: budget hours per day, week or month, per group.



A time budget over one period. Each profile has a time weight; the group's budget is spent at the weighted rate of its work on pools. Weights are illustrative.

Weights say what matters

Each profile has a time weight: the budget hours one hour on that profile costs. Premium GPUs get a higher weight, so they draw a budget down faster. The default is 1. A weight of 0 exempts a profile, such as CPU-only work, which keeps running even when a budget is used up.

Weight per task, not per GPU

The weight applies to each task on the profile, whatever its GPU count or fraction. Set the weight for the whole profile: a profile with several premium GPUs carries a weight that reflects all of them.

Charged for holding capacity

The meter starts when the engine places a task on a pool, including time waiting there for a pod or an autoscaled node, because the capacity is held. Time while the engine is disabled is not charged.

A start check, not a surprise stop

New work starts only if the remaining budget lasts the low watermark, 15 minutes by default, at the group's combined rate with the new task. A small top-up does not start work that would be stopped minutes later. When none of the group's work is running, one task can start even if the budget is smaller than its need.

Used up: work pauses in order

No new work starts from the group. Its running work that spends budget is preempted and goes back to the front of its compute resource, newest first, so the original order holds. Work resumes at the next period or when an admin raises the budget. Use can run slightly over by the time it takes preempted work to stop.

Calendar periods

Periods follow the calendar in UTC: a day from midnight, a week from Monday, a month from the 1st. Unused hours do not roll over.

Opt-in per policy

Policies without a budget keep reservation and limit only, so always-on work such as a model endpoint can run under a policy that is never stopped for budget.

Seen before it hits

Each policy shows hours used and left in the period, the burn rate and when the budget is projected to run out. Queues of a group over budget are marked, and each preempted task carries the reason. Alerts near the limit can be enabled for admins and the group.

06

GPU sizes and sharing

Profiles decide the unit of allocation. A profile can hand out a fraction of a GPU, one GPU or several, and a task's size can come from the task itself.

Fractions

Sessions, small endpoints and experiments can take a fraction of a GPU. On Kubernetes, fractions are memory-capped GPU shares or MIG slices; on bare metal, memory-capped containers in Docker mode.

Task size set by the task

A profile can read a task's size from a task parameter, so one profile covers tasks of different sizes and each counts its own GPUs against the reservation and limit. A default applies when the task sets none.

Sizes by capability

A profile can ask for a capability, such as GPU memory, rather than a GPU model, and list pools from the newest hardware down.

Multi-node work

Large training runs span several nodes on Kubernetes, or use the HPC scheduler's own multi-node placement on Slurm, PBS or LSF.

The GPU as a Service Reference Architecture covers how each kind of work, interactive and non-interactive, uses GPU sizes.

07

Placement and cost

Each profile lists its pools in order. The engine fills them in that order, so the cheapest suitable capacity is used first and cloud capacity is a deliberate step. Reordering a profile's pools changes placement for every channel at once.

Owned capacity first

On-prem Kubernetes → Bare metal → Cloud

The default. Owned GPUs fill first; cloud capacity is added only when they are full and released when idle.

Burn down a commitment

Cloud reserved capacity → On-prem Kubernetes

Where a cloud commitment is prepaid, the cloud pool goes first so the commitment is used before owned GPUs.

HPC first

HPC cluster → On-prem Kubernetes → Neocloud

Batch training uses the HPC allocation first, with Kubernetes and neocloud capacity behind it.

Sovereign by policy

In-country pools only

A group whose data must stay in-country gets profiles that list in-country pools only. Work for that group cannot run anywhere else.

Cloud pools grow and shrink with demand. An autoscaler instance adds VMs, spot first with on-demand as a fallback, up to a maximum per type, and removes idle VMs after a timeout; managed Kubernetes uses the cluster's own node autoscaler. The Cloud Autoscaler Reference Architecture covers both.

08

What a developer sees

Developers and researchers request a GPU size from the channel they already use. They do not need to know where the work runs.

MOMENT	WHAT HAPPENS	WHAT THE DEVELOPER SEES
Request	The task lands on the group's compute resource for that size	The task queued, with its place in line
Within the reservation	Placed on the next pass	The task running, usually within seconds of capacity being free
Above the reservation	Placed on idle GPUs up to the limit	The task running on borrowed capacity
Reclaimed	Preempted and requeued at the front	The task back in the queue with the reason, then running again
Budget used up	Preempted or held until the reset or a raised budget	The compute resource marked over budget, with the reset time
Session left idle	Stopped by the session's idle limits on time, CPU, GPU or network	The session stopped, its GPUs returned to the pool

09

Visibility and governance

Every allocation decision and every GPU-hour is recorded in the control plane, so the same facts serve operations, budget reviews and capacity planning.

Orchestration dashboard

Available and used GPUs and workers in every pool and environment, grouped by group, with use over time and an event log.

Usage ledger

GPU-hours per user, project and group, priced per GPU type, exported by API for showback, chargeback and external billing.

Policy view

Each group's use against its reservation and limit, its time budget used and left, burn rate and projected run-out.

Audit trail

Each task records who started it, on which compute resource and pool it ran, and why it was preempted, if it was.

Model calls are governed the same way through the LLM gateway, with wallets and token quotas per project and user; the LLM Gateway and Tokenomics Reference Architecture covers them.

10

Patterns by organisation

HPC and research labs

Research groups or projects get allocation hours per month as time budgets, the unit research computing already uses. Reservations protect each group's guaranteed share; idle HPC and Kubernetes capacity is lent across groups. Weights charge the newest GPUs at a higher rate.

Enterprise AI platform

Business units get reservations sized to their commitments and limits that let them use idle GPUs. Time budgets cap spend per month on premium GPUs. Model endpoints run under policies without a budget, so production serving is never stopped for budget.

CIO and IT leadership

One estate, on-prem and cloud, governed by one set of rules. Use of owned GPUs rises because idle capacity is lent; cloud spend follows pool order and autoscaler limits; GPU-hours are attributed to every business unit.

Platform teams serving many tenants

Each tenant gets its own share of the estate and sets policies for its own groups inside it. The Multi-Tenant Reference Architecture covers tenants.

11

Design considerations

Size reservations to capacity

Reservations across groups should fit the declared capacity of the pools behind them, or a guarantee cannot be met. Leave unreserved capacity for lending.

Use limits for elasticity, budgets for the total

A generous limit lets groups use idle GPUs; a time budget keeps the total over a period in check.

Weight by value, not by count

Weights apply per task on a profile. Set a profile's weight for its whole GPU size, and weight premium GPUs higher.

Exempt what must not stop

Give always-on work, such as model endpoints, a policy without a budget or a profile with weight 0.

Checkpoint long runs

Borrowed work can be reclaimed and budgeted work requeued. Training code that checkpoints loses little when it is preempted.

Keep declared capacity accurate

"Full" is the declared capacity of a pool, not probed hardware. Update it when hardware changes.

Plan for UTC periods

Budget periods follow the UTC calendar. A group in another time zone sees its reset at the UTC boundary.

Change budgets mid-period with care

Raising a budget releases waiting work at once. Lowering it below what was spent stops the group's budgeted work for the rest of the period.

Check the HPC submitter version

On Slurm, PBS and LSF pools, reservations, limits and time budgets count a task only while the HPC submitter keeps it allocated. Confirm this during acceptance testing.

12

Glossary

Compute policy engine	The part of the control plane that allocates GPUs about once a second: fills reservations, reclaims borrowed GPUs and lends idle ones.
Compute resource	What users and agents send work to. It joins one group's resource policy to one profile.
Resource policy	A group's rules: a reservation, a limit and an optional time budget.
Reservation	The GPUs a group is guaranteed.
Limit	The most GPUs a group can use at once, including borrowed GPUs.
Time budget	Budget hours a group may spend per day, week or month.
Time weight	The budget hours that one hour of a task on a profile costs. 0 exempts the profile.
Low watermark	How long the remaining budget must last, at the group's rate, for new work to start. 15 minutes by default.
Profile	A GPU size, whole GPUs or a fraction, with a time weight and an ordered list of pools.
Pool	A set of nodes in one place with a declared capacity, served by Klique worker daemons.
Klique worker daemon	The worker in each pool that pulls work from the control plane over outbound HTTPS and starts it.
Preemption	Stopping placed work and returning it to the front of its compute resource, to reclaim a reservation or when a budget is used up.
Usage ledger	The record of GPU-hours per user, project and group, priced per GPU type.

Klique AI · Resource allocation and cost governance reference architecture · one GPU estate shared under

reservations, limits and time budgets. Groups, weights and pool orders are illustrative.