ChatGPT

Software Stack and Multi-Tenancy for a GPU Supercluster (2025)

Software Stack Components for GPU Clusters

Building a multi-tenant GPU cluster requires a full software stack from bare-metal OS up to user-facing ML platforms. At the base is the **operating system** – typically a 64-bit Linux distribution optimized for HPC/AI (common choices are Rocky Linux/Red Hat or Ubuntu). The OS provides kernel support for NVIDIA drivers, high-speed interconnects (InfiniBand/RDMA), and filesystems. On top of the OS, the **GPU driver and CUDA libraries** must be installed (often via NVIDIA's CUDA toolkit or container images). Many clusters use container runtimes (Docker/Containerd) to deliver user workloads in isolated environments. NVIDIA's NGC containers (part of **NVIDIA AI Enterprise**) bundle drivers and AI frameworks for consistency [1].

For cluster-wide orchestration, **Kubernetes** has become popular for AI/ML workloads due to its cloud-native approach and rich ecosystem. Kubernetes Device Plugins (like NVIDIA's GPU Operator) handle GPU scheduling on each node. Alternative or complementary to Kubernetes, traditional HPC batch schedulers (e.g. **Slurm**) are still used for training jobs, offering high throughput on multi-node GPU jobs with bare-metal performance. In modern converged clusters, it's common to deploy *both* Kubernetes and Slurm side-by-side – e.g. Slurm for batch jobs and Kubernetes for services or interactive workloads [2] [3]. Each approach can be toggled per node or workload using deployment profiles (for example, a cluster could dedicate some nodes to a Slurm partition and others to a Kubernetes cluster, or even have nodes serve both roles with containerized Slurm jobs).

At the software stack's higher layers are the **AI/ML libraries and frameworks**. NVIDIA's AI stack (part of **NVIDIA AI Enterprise (NVAIE)**) provides optimized builds of TensorFlow, PyTorch, RAPIDS, etc., plus GPU-accelerated libraries (cuDNN, NCCL, TensorRT, etc.). NVAIE is a **cloud-native, end-to-end AI software platform** that accelerates data science pipelines with enterprise support [4]. Notably, NVAIE is a licensed suite (per GPU) that ensures compatibility and support – it **requires a license for each GPU** on which it runs [5] (although newer GPUs often include NVAIE subscriptions for a few years – e.g. each NVIDIA H100 or H200 NVL GPU comes with a 5-year AI Enterprise license included [6]). This provides the cluster with all necessary AI software, drivers, and tools under enterprise support. Alternative open-source stacks (without NVAIE) can be used too (e.g. relying on community CUDA and open-source ML frameworks), but organizations often opt for NVAIE to get **validated software and support** for production clusters.

Networking and storage software are also critical. GPU clusters at scale use high-speed fabrics (NVIDIA Quantum-2 InfiniBand or 200Gb+ Ethernet with RDMA) – typically managed by drivers and SDKs (e.g. RDMA core, NVIDIA Fabric Manager for NVLink/NVSwitch). Storage may involve parallel filesystems or object stores accessible across tenants. The software stack should include monitoring and management tools as well: for example, cluster metrics collectors (Prometheus, node exporter, DCGM for GPU metrics) and a visualization UI (Grafana dashboards, or vendor-specific GUIs).

In summary, the core stack on each node is **bare-metal Linux + NVIDIA drivers + container runtime + Kubernetes/Slurm agents + AI libraries**. This provides a foundation that avoids heavy hypervisors – the cluster runs mostly on bare metal to maximize performance (the user explicitly wants *no OpenStack* due to ~10% virtualization overhead). Instead of VMs, isolation is achieved at the container and hardware level (using features like NVIDIA Multi-Instance GPU and Linux containers, discussed below).

Automation Tools for Deployment and Lifecycle Management

Deploying and managing a massive GPU cluster (potentially many racks scaling to hundreds or thousands of GPUs) demands robust automation. Key tasks include bare-metal provisioning, firmware/BIOS updates, OS installation, network configuration, and installing the cluster software (drivers, Kubernetes or Slurm, etc.). Several frameworks exist to automate **initial deployment** and **full lifecycle management** of such clusters:

- **Dell** Omnia*** – *an open-source toolkit (developed by Dell Technologies HPC/AI team) designed to automate end-to-end cluster setup. Omnia uses* *Ansible playbooks** to provision the OS, configure networking, install GPU drivers, and deploy cluster services like Slurm or Kubernetes [7]. It bundles roles for common HPC/AI components (e.g. CUDA libraries, InfiniBand, Lustre, etc.) and can deploy *either* Slurm batch scheduler, *or* a Kubernetes cluster (or both) on the nodes [2]. Omnia essentially provides "**clusters-as-code**": you describe in config which nodes should be in the Slurm partition, which as K8s masters/workers, etc., and Omnia applies the corresponding configuration. It even automates tuning and library installation for popular ML frameworks and can deploy monitoring (it integrates with Grafana for telemetry, even pulling metrics from node iDRAC hardware sensors) [8]. Omnia is **Apache 2.0 licensed (fully open-source)**, so you can fork or extend it freely [9]. It was built specifically to simplify mixed AI/HPC clusters – letting admins "compose" and "recompose" resources easily via templates instead of re-imaging nodes for each change [10]. For example, Omnia lets you repurpose a set of nodes from an HPC MPI cluster to an AI training Kubernetes cluster by running a playbook, rather than manually reinstalling everything. This flexibility is a big plus for evolving workloads. Dell offers Omnia as part of their validated HPC/AI solutions, and while the tool is community-supported, **Dell provides support services for Omnia** if you have a Dell solution (since it's used in Dell's AI/HPC Ready Solutions). In short, Omnia covers bare-metal provisioning through configuration: it abstracts away manual steps and reduces deployment time "from weeks to hours" by imprinting the needed software stack onto each server based on its role [11] [12].

- **NVIDIA Deployment Tools** – NVIDIA provides software for cluster deployment as well. In the past this was often via the **DeepOps** project (an NVIDIA-maintained open-source playbook collection similar to Omnia for deploying GPU clusters with Slurm or Kubernetes). In enterprise offerings, NVIDIA now emphasizes **NVIDIA Base Command Manager (BCM)** and related tools. **Base Command Manager** is effectively the "operating system" for DGX clusters – it automates provisioning of AI clusters, monitoring, and job scheduling at scale [13]. Notably, as of mid-2025 NVIDIA has made Base Command Manager **free to use** (for clusters of any size, supporting up to 8 GPUs per node) [14] [15]. This means you can deploy NVIDIA's cluster management software without license fees, and then optionally buy enterprise support if needed. BCM provides a web UI and APIs to manage cluster resources, schedule jobs (including containerized workloads on bare metal), monitor GPU utilization, etc. – it's designed for multi-user environments (a single cluster can securely host multiple users or teams) and supports both containerized and non-container workloads [16]. Base Command can integrate with Kubernetes (for example, it works with the NVIDIA GPU Operator

on K8s) as well as manage bare-metal Slurm jobs. NVIDIA has also introduced **NVIDIA Mission Control**, an advanced orchestration platform that *builds on Base Command* for "AI factories." Mission Control adds automation and "full-stack intelligence" to manage everything from **developers' workloads to the underlying infrastructure and even facilities (power/cooling)** with hyperscale efficiency [17]. Essentially, Mission Control is aimed at very large deployments (think multi-rack superpods) to **optimize utilization and automate the AI pipeline end-to-end**. While Base Command Manager is now free (with community support) [18], Mission Control is a higher-tier product (likely licensed or bundled with NVIDIA's ultra-scale systems). Both are part of NVIDIA's effort to provide turnkey software for multi-tenant AI clusters – an admin could deploy Base Command to handle user management, scheduling, and telemetry instead of assembling open-source tools manually. If you purchase **NVIDIA AI Enterprise** licenses, those also entitle you to support for Base Command and related operators [19], so NVIDIA's software stack is integrated.

- **Spectro Cloud Palette** – a commercial platform from Spectro Cloud that focuses on Kubernetes cluster management with fine-grained control. Palette can **provision Kubernetes on bare metal** (as well as on cloud VMs or edge devices) and manage the full lifecycle (upgrades, day-2 operations) through a single pane. It's relevant here because Spectro Cloud has an **"AI stack" integration**: for example, they have a packaged profile to deploy NVIDIA's GPU Operator and necessary drivers on clusters [20]. Palette supports multi-tenant use cases by allowing either multiple clusters (one per tenant or project) or **"virtual clusters"** (nested K8s clusters sharing a physical cluster). In fact, Spectro Cloud offers **"virtual clusters"** as a feature, which can isolate tenants at the Kubernetes API level while sharing underlying nodes [21]. The platform itself can run in a multi-tenant SaaS mode or on-prem. Under the hood, Spectro Cloud leverages technologies like Cluster API for provisioning and provides a catalog of "packs" (addons) – e.g. an **"AI Pack"** that includes the NVIDIA GPU Operator, drivers, and perhaps Kubeflow components [22]. The **openness** of Spectro Cloud: it's not open-source; it's a proprietary solution (with a free trial likely, but source code is not open). However, it largely uses open standards (Kubernetes, Cluster API) so you're not locked into custom orchestration formats. You wouldn't be forking Spectro's code, but you could export cluster definitions (YAML, etc.) and reimplement elsewhere if needed. Spectro's **pricing model** is usage-based – for example, Palette Enterprise edition is billed by managed CPU core-hours (kilo-core hour units) with no minimum [23]. They also have a **per-host annual pricing option for edge deployments** [24]. This might be attractive if you want a supported, ready-to-go Kubernetes management layer rather than building your own with pure open-source tools.

- **Canonical MAAS + LXD/Incus MicroCloud** – As an alternative to Kubernetes, the user mentioned Canonical's **MicroCloud**. Canonical's MAAS (Metal-as-a-Service) can automate the bare-metal provisioning (firmware, OS installation) on large clusters, and then **LXD/Incus** can be used to orchestrate workloads in lightweight VMs or system containers. **LXD/Incus** essentially provides VM-like isolation with containers – you can spin up a container running an OS (with its own init, user space) on each node with near bare-metal performance. A *MicroCloud* is a small cluster running LXD containers that can be managed somewhat like a cloud. This approach could allow each tenant to get an isolated OS environment (more isolated than just a Docker container, since they have separate init and can even run different distros) with minimal overhead (LXD containers share the host kernel, so overhead is very low, much less than a full VM hypervisor). Incus is the new community fork of LXD (after Canonical changed LXD's license). Using LXD/Incus might be advantageous if you want to avoid the complexity of Kubernetes for certain scenarios – e.g. you could script LXD to launch a container with X GPUs passed-through for a tenant's long-running job. However, LXD on its own

doesn't provide advanced scheduling or multi-tenant resource fairness – it's more akin to a manual IaaS. In practice, many organizations would layer Kubernetes on top for scheduling even if using LXD (for example, running Kubernetes inside LXD VMs, or using LXD as the container runtime for K8s). So, while **MicroCloud** can help quickly provision a mini-cloud environment, for *massive scale* and rich multi-tenant services, Kubernetes (with its ecosystem) is still the go-to solution in 2025.

- **Other vendor tools**: **Lenovo** provides its **Lenovo Intelligent Computing Orchestration (LiCO)** software, which is a **cluster management and user portal solution for HPC/AI**. LiCO isn't a provisioning tool per se, but rather sits on top of an existing cluster (with Slurm or K8s) to simplify usage. It provides a web UI for users to submit jobs (including AI jobs like TensorFlow training) via templates, manage data, and even handles multi-tenant concerns like user group management and chargeback accounting [25] [26]. For example, LiCO supports defining **billing groups and fees, user quotas, and tracking usage for chargeback** – features critical in a GPUaaS environment [26]. It also provides end-to-end training workflows (with prebuilt templates for common AI tasks), and manages container images for users (administrators can supply base AI container images, and users can run jobs in those or custom images) [27]. LiCO can operate with Slurm as the backend scheduler (currently Slurm is the supported engine for job dispatch) and can also interface with Kubernetes clusters for AI workloads [28] [29]. Essentially, LiCO is **Lenovo's answer for multi-tenant AI clusters**, abstracting away the complexity of the underlying cluster and giving researchers or data scientists a friendly portal to request resources, launch jobs, and monitor progress [30] [31]. It's a proprietary solution (licensed with Lenovo – often bundled with their ThinkSystem clusters or sold as an add-on). Lenovo also has reference **EveryScale HPC & AI stacks** that include LiCO, Slurm, Kubernetes, etc., as part of their turnkey offerings [32].

Supermicro focuses primarily on hardware integration but also offers management software. Their new large clusters (for example, a rack-scale "SuperCluster" with NVIDIA Blackwell GPUs and Grace CPUs) come with a **"SuperCloud Composer"** software for infrastructure management [33]. SuperCloud Composer provides tools to monitor and optimize at the rack level – especially important for things like liquid cooling loops, power, and thermals in high-density GPU racks [34] [33]. It's more about hardware orchestration (fans, pumps, BIOS, firmware updates, etc.) and less about end-user job scheduling. For actual workload management, Supermicro leans on partners: their systems are **validated for NVIDIA AI Enterprise** and they tout "native support" for NVAIE's software platform [35]. In practice, if you buy a Supermicro GPU cluster, you might use NVIDIA's Base Command or a Kubernetes/Slurm stack on it, with SuperCloud Composer ensuring the hardware stays healthy. Supermicro's expertise is delivering the cluster fully integrated (with networking, cooling, etc. ready to go), plus providing the management layer for the physical infrastructure [36]. You would still choose a software stack (could be Omnia + Slurm, or NVAIE + Kubernetes, etc.) for the tenant-facing side.

Other players like **HPE** have their **Performance Cluster Manager (HPCM)** – a derivative of the old SGI cluster tools – and **Bright Cluster Manager** (which NVIDIA acquired in 2022) was another common tool for deploying HPC clusters (Bright could provision and manage clusters similarly to Omnia, with a nice GUI, and handle multi-tenant scheduling via Slurm/K8s integration). NVIDIA has folded a lot of Bright's functionality into its current offerings (Bright's tech underpins some of Base Command and perhaps Mission Control). **Open-source tools** like **xCAT, Warewulf, and Cobbler** have historically been used for bare-metal provisioning in HPC; these can still be used (Omnia, for instance, uses warewulf internally for stateless provisioning of compute nodes in some configurations). And for configuration management, Ansible is dominant, but others might use Puppet or Terraform (for example, using Terraform with vendor Redfish

APIs to orchestrate server provisioning). The key point is that **full automation from firmware to OS to cluster services is achievable with modern tools** – critical for rapidly scaling the cluster or replacing failed nodes with minimal downtime. The cluster should be treated as code: version-controlled playbooks/ manifests define what a new node or tenant environment looks like, so you can reproduce it consistently.

Multi-Tenancy and GPU Sharing Strategies

A central requirement is **stacking as many tenants as possible onto the hardware** (e.g. those powerful "GB200/300 NVL72" GPU systems) *without compromising security or performance*. In 2025, NVIDIA's hardware and software offer strong solutions for safe multi-tenancy on GPUs:

- **NVIDIA Multi-Instance GPU (MIG)** – This is a feature of NVIDIA A100, H100, and newer data center GPUs that allows a single physical GPU to be partitioned into up to **7 isolated instances** [37]. Each MIG instance has dedicated compute cores, its own high-bandwidth memory subset, cache, and a proportionate share of memory bandwidth. The instances are **fully isolated** at the hardware level, with performance QoS guarantees and memory protection between tenants [37]. For example, one H100 80GB can be split into 7 MIG slices (~10GB each) or other size combinations. MIG essentially lets multiple tenants run on one GPU concurrently as if each had a smaller GPU, *with negligible overhead* and strong isolation (NVIDIA describes MIG as providing the same isolation guarantees as their virtual GPU (vGPU) technology) [38]. **MIG is ideal for inference or smaller training jobs** – you could have 7 users each running a model on 1/7th of a GPU, securely. MIG instances can also be dynamically reconfigured; admins can adjust how a GPU is partitioned (e.g. fewer larger instances vs more smaller) to match workload demand, without rebooting the host [39]. This dynamic ability means your service could offer different "GPU slice" sizes in a catalog (like small = 10GB MIG, medium = 20GB MIG, etc.) and change the allocation as needed. Since MIG is available only on certain GPU models, your mention of "GB200/300 NVL72" suggests very high-end systems (possibly Grace+Blackwell based HGX with NVLink clusters). These presumably will continue MIG or similar partitioning support (NVIDIA Blackwell is expected to support multi-instance or advanced slicing). Using MIG in a multi-tenant cluster provides **bare-metal performance** for each tenant's slice with isolation, making it far more efficient than giving each tenant a whole GPU if they don't need it.

- **GPU Virtualization (vGPU)** – NVIDIA's older approach for sharing GPUs uses software virtual GPU (vGPU) managers on hypervisors like VMware or Linux KVM. This allows multiple VMs to each see a virtual NVIDIA GPU (backed by time-slicing or partial allocation of a physical GPU). vGPU can offer smaller granularity (e.g. frame-buffer partitioning into many slices) and is widely used in VDI or cloud GPU offerings. **However**, vGPU for data center (NVIDIA Grid/Virtual Compute Server) requires a per-GPU licensing fee and typically runs on a hypervisor (ESXi, etc.), which adds overhead and complexity. Given the user's constraints (no heavy virtualization), vGPU in the traditional sense is not ideal – it would introduce similar overheads to OpenStack/VMware. MIG is a preferable form of "hardware vGPU" on bare metal. (One could technically run vGPU on bare-metal containers using NVIDIA GPU Operator with vGPU mode, but that still requires licenses and doesn't eliminate the overhead of scheduling context switches.)

- **Multi-Process Service (MPS)** – NVIDIA MPS is a software feature that lets multiple CUDA processes share a GPU more cooperatively (primarily to minimize context switching overhead). It doesn't enforce memory isolation (processes could potentially read each other's GPU memory if not container-isolated) and is meant more for single-user multi-process scenarios. In a multi-tenant

environment with untrusted tenants, MPS alone isn't safe. So we wouldn't rely on MPS for secure multi-tenancy, especially when MIG exists.

Container Isolation and OS Security: In a multi-tenant *containerized* cluster (Kubernetes or LXD), each tenant's workload would at least be isolated by Linux namespaces and cgroups. That provides process and resource isolation, but not as strong as VM isolation. MIG strengthens it at the hardware level for GPUs, and one can also enhance container isolation via techniques like **Kubernetes PSP/OPA policies**, **seccomp** profiles, etc., to limit what a container can do. If tenants are truly independent (e.g. external customers), you might consider using **Kata Containers** or similar "sandboxed containers", which run each container inside a lightweight VM (Firecracker or QEMU) automatically. Kata Containers integrate with Kubernetes, giving nearly VM-level isolation per pod, while adding only a small overhead (~5-10% typically). This can alleviate concerns that one tenant could exploit the shared kernel – important in a GPUaaS scenario with potentially malicious users. The trade-off is some performance hit, but far less than a full cloud stack like OpenStack. Many GPU clouds, for example, rely on Kubernetes + Kata or gVisor for isolation at scale.

Networking isolation: The software stack should also enforce network separation between tenants – e.g. Kubernetes network policies can silo tenant namespaces, or if using multiple clusters, ensure proper VLAN/VXLAN segmentation per tenant. In a multi-tenant cluster, it's wise to integrate a CNI that supports network isolation and maybe even rate limiting (to prevent one tenant from saturating bandwidth).

Secure Multi-Tenancy in Practice: The best practice is often to combine these layers: *hardware isolation (MIG) + orchestration-level isolation (separate Kubernetes Namespace or even separate cluster per tenant) + strong policy enforcement (RBAC, network, runtime security)*. For example, **NVIDIA's recommended approach** for multi-tenant Kubernetes uses their device plugin to advertise MIG partitions as distinct schedulable resources, so that Kubernetes can schedule GPU slices to different pods/tenants. Each pod sees only its MIG device (e.g. `nvidia.com/mig-3g.20gb` resource) and cannot access others. The **overhead of MIG is minimal** – MIG partitions have virtually no performance penalty versus bare metal (they are hard-partitioned, not time-shared), aside from not being able to use *more* than their share of the GPU. That makes it very efficient to maximize utilization.

One caveat: If tenants require *whole GPUs* for large training jobs, you won't be "stacking" those particular jobs on the same GPU – but you still stack them on the same cluster by allocating different GPUs to each. In that case, the scheduler (Slurm or K8s) ensures each job gets exclusive GPUs. The multi-tenancy then is at node or cluster level, not within one GPU. But even then, the software stack must isolate other resources (CPU, memory, NIC bandwidth) between jobs; Kubernetes does this with cgroups and quotas, Slurm does it by cgroup confinement as well.

Finally, **full cluster multi-tenancy** means isolating not just GPUs but also ensuring one tenant's processes can't snoop on another's data in RAM or interfere with control planes. Running separate Kubernetes clusters per tenant is the strongest isolation (very similar to how public cloud works: each customer in Kubernetes gets their own cluster or virtual cluster). However, managing hundreds of separate clusters is complex – instead, a single cluster with multi-namespace multi-tenancy can be used, with proper policies. Projects like Kubernetes Multi-Tenancy Working Group have patterns for this (like hierarchical namespaces, etc.), and Spectro Cloud's **virtual clusters** approach essentially gives each tenant a Kubernetes API of their own backed by a shared cluster.

No OpenStack? The user specifically notes not using OpenStack – indeed OpenStack would provide multi-tenant VM isolation, but at a cost. OpenStack's own services consume significant overhead and VMs mean a hypervisor layer (roughly ~10% performance cost is a rule of thumb). By avoiding that and going direct to bare metal + containers, we save that overhead and can pack more performance per GPU. We just have to carefully implement isolation by other means, as described.

Tenant Service Models and Self-Service Workflows

To operate this as a **GPU-as-a-Service (GPUaaS)** platform for potentially *hundreds or thousands of users*, you need a way to let tenants request and consume resources without manual admin work for each. This is where a **service catalog and automated MLOps workflows** come in. Essentially, you want predefined "service tiers" or templates that a tenant can choose from – and behind the scenes, automation provisions the necessary containers, data pipelines, and so on.

Possible service tiers that could be offered in such an environment (inspired by existing GPU cloud offerings):

- **Interactive AI Workspaces** – e.g. a **JupyterLab/VSCode Notebook server** with one or more GPUs attached. This could be a tier for data scientists to do explorations. You might offer different sizes: e.g. *Small Notebook* (1 MIG slice, 16 GB RAM), *Large Notebook* (1 full GPU, 64 GB RAM), etc. This is analogous to services like Google Colab or Azure ML Notebooks. The software to enable this could be something like **JupyterHub on Kubernetes** (multi-tenant Jupyter service), possibly integrated with authentication so that each user gets a container spawned with appropriate resources. Kubernetes operators (like the Kubeflow Notebook Controller) exist to manage that.

- **Batch Training Jobs** – Tenants can submit a training job (maybe via Git repository or via container image) and specify how many GPUs, what framework, etc. The system would schedule it through Slurm or Kubernetes Jobs. You could have **predefined training pipelines** ("train this model on provided dataset") so that a user just triggers it with minimal fuss. For example, **Kubeflow Pipelines** or **Argo Workflows** can define repeatable ML pipelines (data prep -> training -> validation -> deploy). A tenant could select a pipeline from a catalog (like "Train ResNet on new dataset") and the automation runs it on the cluster. This provides consistency and saves them reinventing the wheel each time. NVIDIA offers some **AI workflow scripts** (NGC has pre-trained model scripts, etc.), and Dell mentions **"Accelerated AI templates"** in their solutions [40] – those could be integrated so a tenant can spin up a known workflow (for instance, fine-tune an LLM on their data).

- **Inference Services** – Once models are trained, tenants may want to deploy them as APIs. A service tier could be *Inference Endpoint*: the tenant provides a model (or chooses one from a library), and the platform deploys it on a GPU with an endpoint (maybe using NVIDIA Triton Inference Server under the hood). Multi-tenant inference is tricky (you want high utilization, so perhaps multiple models per GPU, using MIG or batching). But an orchestrator like Kubernetes makes it feasible: for example, each model deployment is a pod or set of pods, and an API gateway routes requests. You'd also implement autoscaling (spinning up more pods if load increases). Many GPU cloud providers (AWS, GCP) have this – AWS has **SageMaker Endpoints**, GCP has **Vertex AI Predictions** – on-prem, one could use **KFServing (KServe)** or Triton to similar effect.

- **Data Science "Playground"** – possibly an environment with data processing tools (Spark or Dask) integrated with GPUs. If you expect tenants doing data processing, you might offer a service like *GPU-accelerated Spark cluster on demand*. NVIDIA's RAPIDS + Spark can accelerate ETL on GPUs. This could be either an interactive service or a batch job service.

- **DevOps/CI-CD Integration** – You mentioned tenants triggering a CI pipeline towards the infra via API. A possible implementation: tenants integrate their GitLab/Jenkins with your API such that pushing new training code triggers a pipeline that calls, say, an **Argo Workflow** on your cluster. The Argo workflow could build a container, run the training, then push the results. This way, the cluster offers "continuous training" capability. To do this safely, you'd expose certain APIs or use a token system so tenants only trigger their own workflows. This approach aligns with the concept of an **"AI Factory"** – where code and data come in, models come out, with as much automation as possible.

To support all that, you need a layer above raw Kubernetes/Slurm. As mentioned, **Lenovo LiCO** is one example that already provides a lot of this: job templates, a web GUI to submit jobs, monitor them, and manage user accounts/quotas [41] [42]. LiCO even has built-in billing and reporting, which is incredibly relevant (it can track usage per billing account and supports chargeback). Using something like LiCO or Open OnDemand (an open-source HPC portal) could save development effort. Alternatively, **Kubeflow** is an open-source project aimed at offering a cohesive ML platform on Kubernetes – it includes multi-user Jupyter notebooks, pipeline orchestration, hyperparameter tuning, model serving, etc. Kubeflow can be a bit heavy to set up, but it's designed for exactly multi-tenant ML use-cases on Kubernetes. Some organizations also adopt **MLflow** for experiment tracking and model registry, so tenants can log their runs and save models, then later deploy them.

Service Catalog Interface: It's a good idea to present tenants with a self-service **catalog/portal**. This could be a web UI where they select what they want (Notebook, Training Job, etc.), or a CLI/API for advanced users. For instance, a user could call `POST /api/notebooks` with parameters and the automation (through Kubernetes operators) provisions it. If you implement it as simply as giving each tenant a Kubernetes namespace and some Helm charts for common services, that could work too (the tenant runs `helm install notebook` in their namespace, etc., if they are savvy). But a curated portal is more user-friendly.

Metering and Quotas: Multi-tenant implies you must have **quotas** (to prevent one user from hogging all GPUs) and **metering for chargeback**. In Kubernetes, **ResourceQuota** can cap how many GPUs (or CPU, memory, etc.) each tenant namespace can use at a time. On the monitoring side, tools like **Kubecost** can track usage of resources per namespace and even assign dollar values. For Slurm, there's an accounting database (Slurmdbd) that records each job's resource usage; you can extract per-user or per-account GPU-hours. Many HPC centers charge by the GPU-hour or CPU-hour similarly. LiCO, as mentioned, supports "user top-ups and chargebacks" natively [26], meaning it has the concept of billing units that users consume as they run jobs. That might be very useful to adapt – you could set a rate (say $X per GPU-hour) and LiCO would deduct from a tenant's balance as they run jobs, for example.

Pre-defined MLOps flows: The question specifically suggests having *pre-defined service flows* (MLOps flows) that can be triggered from a catalog or via CI pipeline. Let's consider an example flow: "Train and Deploy Model". This could correspond to a Jenkins pipeline that, when triggered, does: checkout code -> build container -> submit training job to cluster -> wait for completion -> on success, register model and deploy to inference server. You could offer that as a one-click in a portal. Implementing this might involve Jenkins or

GitLab CI on the back-end integrated with cluster APIs. There are also specialized MLOps platforms (e.g. **Azure ML** or **DGX Foundry** style products) that integrate these steps, but if building in-house, using open-source CI plus Kubernetes jobs is a common approach.

Comparing to other GPU cloud offerings: It's useful to see what **service tiers** existing GPU cloud providers offer, as inspiration:

- **NVIDIA DGX Cloud** (and older NVIDIA Base Command Service) – NVIDIA's own cloud offering (hosted through partners like Oracle) provides essentially leased multi-GPU nodes that users can access, with Base Command to schedule jobs. They focus on large dedicated instances rather than many small tenants, but they do allow multiple users on a cluster via their scheduler. The services include job scheduling, interactive development sessions, and managed storage. One notable thing NVIDIA provides are **pre-trained models and AI workflows** in NGC, which a tenant can deploy easily. For instance, a catalog of NVIDIA **AI Enterprise microservices** (like Jarvis speech service, etc.) is available to deploy on the cluster [43] – that could be something you mirror, offering tenants ready-made models (so they don't even need to train from scratch if a pre-trained model suffices).

- **AWS** – Amazon's GPU offerings include raw VMs (EC2 P4d/P5 instances with A100/H100), as well as higher-level services. **Amazon SageMaker** is their managed ML platform – it provides notebook instances, training jobs (on a fully managed Kubernetes behind the scenes), and inference endpoints. They have **SageMaker JumpStart** which is a model zoo that users can pick models from and deploy quickly. AWS's multi-tenancy is at the account level (each user gets an AWS account or project), but within SageMaker they handle user isolation. They also offer features like **SageMaker Pipelines** (CI/CD for ML) and monitoring for deployed endpoints. The key takeaway from AWS is the convenience of a one-stop solution – which you might replicate via Kubeflow or LiCO.

- **Google Cloud** – GCP has **Vertex AI**, similar to SageMaker. It provides Training Pipelines, Notebook services, and Prediction services. Google also offers **fractional GPUs** in Google Kubernetes Engine (GKE): for example, you can allocate 0.5 of a GPU (they use time-slicing or MIG under the hood for that). They are essentially doing what we described – using MIG or vGPU to let smaller pods share a GPU. This indicates that *the concept is proven*: major clouds allow sub-GPU billing to maximize utilization.

- **CoreWeave** – This is a prominent specialized GPU cloud provider. CoreWeave runs on Kubernetes and has developed their own scheduler and tooling to allow very flexible GPU allocation. They offer fractions of GPUs (e.g. 1/8th of an A100) and also time-sliced shared GPUs. They've built a reputation for having a wide variety of GPU types and allowing bursting and spot instances. CoreWeave's stack is basically Kubernetes + custom controllers; they even acquired a company that did cloud scheduling for VFX (Conductor). In terms of tenant services, CoreWeave mostly exposes infrastructure (you deploy containers or VMs via their API) rather than a full ML workflow platform. But their selling point is efficiency: by packing multiple customers on GPUs using fractional allocations, they keep utilization high and costs lower. They likely leverage MIG for A100s and perhaps vGPU time-slicing on older GPUs. CoreWeave also open-sourced some of their tooling for fractional GPU scheduling. The success of CoreWeave shows that a *containers + GPU operator + custom scheduler* approach can indeed support multi-tenant GPU cloud at scale.

- **Lambda Cloud** – Lambda (known for on-prem GPU workstations) also has a cloud service. They focus on offering bare-metal GPU instances (no fractional sharing publicly, I believe) and they emphasize simplicity and low cost. Lambda's interesting angle is that they also provide **on-prem cluster solutions**: e.g. the press has a story of *Lambda building AI "factories" with Supermicro Blackwell GPU clusters* [44]. This presumably means Lambda will sell/rent you a whole cluster and possibly manage it. Lambda's software stack for on-prem likely includes their **Lambda Stack** (which is basically Ubuntu + ML frameworks) and perhaps container orchestration. If Lambda is partnering with Supermicro, they might incorporate things like SuperCloud Composer for hardware and maybe Kubernetes for scheduling. It's possible they have a simple management layer for multi-user job submission (but details are scarce publicly). The takeaway is that even smaller vendors recognize the need to provide a *full stack experience* (hardware + software tuned for AI).

- **Paperspace** – Another cloud service (owned by Digital Ocean now) which provides a platform called **Gradient**. Gradient is a web platform where users can spin up Jupyter notebooks, run experiments, and deploy models easily, without dealing with low-level cloud config. It's very much a **data science playground** approach. They likely run on top of a Kubernetes cluster with multi-tenancy. They offer features like collaborative notebooks, dataset versioning, etc. This is relevant because it's exactly the kind of experience an internal platform might want to offer to its users/tenants to drive adoption.

In designing your service tiers, you would likely choose **some combination of these patterns**: e.g. *Dedicated Training* (user gets X GPUs exclusively for a job, possibly scheduled via a queue with fairness), *Shared Interactive* (multiple users share a GPU via MIG for interactive work), and *Managed Inference* (the platform handles scaling/HA of model servers). Each tier can have different pricing.

For example, a **three-tier model** could be: (1) *Development Tier* – shared MIG slices, preemptible (cheaper, best-effort, for experimenting), (2) *Standard Tier* – full GPU or multi-GPU reserved for the tenant's workload, higher cost, (3) *Enterprise Tier* – dedicated nodes or guaranteed QoS with perhaps better support/SLA, for critical training jobs. This is just an illustrative concept – the actual tiers can be whatever aligns with usage patterns. Importantly, since you don't want to manually set up each tenant environment, you'll automate provisioning of their namespaces or virtual clusters, and their access credentials, etc., through onboarding scripts or an "organization creation" workflow.

Licensing and Cost Considerations

Now let's talk about **licenses and costs** for all these software components, and how to estimate TCO. A large GPU cluster has several licensing aspects:

- **NVIDIA AI Enterprise (NVAIE)**: As noted, NVAIE is licensed per GPU. It can be bought as annual subscriptions or perpetual (with support). Pricing isn't usually published openly – you go through partners – but to give an idea, a 5-year subscription for 1 GPU was listed around $18,500 (which is roughly $3,700 per GPU/year) [45]. Enterprise discounts, of course, apply at volume. However, if you're buying the latest NVIDIA GPUs, you might *not need to purchase NVAIE separately*: **NVIDIA now includes NVAIE subscriptions with certain GPUs** – e.g. each H100 PCIe or H100 NVL GPU includes a 5-year AI Enterprise license (and some A800 GPUs include 3-year) [46]. So, if your cluster uses those, the software stack license is effectively bundled (just needs activation). Future Blackwell GPUs (H200 NVL, etc.) also come with 5-year NVAIE [47]. After the included period, you'd need to renew if

you want continued updates/support. If you have older GPUs (say A100s that didn't include this), you'd have to buy licenses or run the open-source stack without enterprise support.

- **NVIDIA Base Command / Mission Control**: Base Command Manager is now free to use (no license fee) [14]. That's great for reducing TCO – you can deploy it and only pay if you want support. Enterprise support for it is included if you have NVAIE, or can be purchased standalone [19]. NVIDIA Mission Control, being newer and more comprehensive, likely comes at additional cost (it's pitched to enterprises building "AI factories" who likely will engage NVIDIA consulting or purchase it as part of a big solution). Details on its licensing aren't public, but one can assume it might be a per-cluster or per-node subscription. For budgeting, if Mission Control is of interest, you'd treat it similar to an enterprise software license line item.

- **Dell Omnia**: Omnia itself is free and open source. There's no licensing fee. Dell would monetize it by selling the hardware and perhaps services. For instance, Dell could offer an **Omnia support contract or deployment service** – e.g. Professional Services to set up the cluster using Omnia, or ProSupport for HPC which covers the environment. If you want to fork/modify Omnia, you can (Apache 2.0 license). So from a cost perspective, Omnia helps avoid having to buy something like Bright Cluster Manager (which used to be a licensed product). **Dell's broader "AI Factory" solutions** might include other software though: Dell partners with NVIDIA, so if you buy a Dell-NVIDIA solution, you might get NVAIE and Base Command bundled. Dell also has **OpenManage** tools for hardware – e.g. **iDRAC Enterprise** licenses (for each server) are usually required for remote management (these often come with the server purchase, or as an added cost of a few hundred $ per server). If using Dell's **Bare Metal Orchestrator** (a tool for automated server provisioning often used in telco), that could be another licensed software, but it's probably not needed if Omnia covers provisioning.

- **Spectro Cloud Palette**: As gleaned, Palette's pricing is based on usage (core-hours). They emphasize "no minimum spend" and basically bill like a cloud service. Without an actual price sheet, let's hypothesize: if you managed a cluster of 100 nodes, each with e.g. 64 CPU cores, that's 6400 cores. If they charged (for example) $0.0001 per core-hour (just a random guess), that'd be $0.64/hr for the whole cluster, about $460/month. But this is speculative – the actual rate could differ. The key is, SpectroCloud is a subscription OPEX model. One would compare this cost to the cost of employing engineers to manage pure open-source Kubernetes; if Palette saves a lot of time, it might be worth it. Since they do custom quotes, you'd request a quote for, say, managing X clusters with Y nodes. If you needed offline (air-gapped) self-hosted use, they might have a flat yearly fee per cluster instead. It's not open-source, so you cannot avoid subscription if you choose them.

- **Lenovo LiCO**: Lenovo likely licenses LiCO per cluster or per site. In the **Lenovo Press product guide**, LiCO is described as a software solution included in their AI offerings. A part number (7S09002BWW) was found in search, suggesting LiCO might be sold as a bundle. It supports the major OSes (RHEL, SLES, Ubuntu) [48], so no additional OS license is embedded. If you buy a Lenovo HPC cluster, LiCO might be included "for free" or a small charge, especially if it helps them sell hardware. If purchasing standalone, you'd contact Lenovo for pricing. Compared to writing your own portal, LiCO could be cost-effective if you already are a Lenovo shop. It's also actively maintained (the admin guide quoted is for LiCO 8.0 in 2025).

- **Operating System**: If you use a community OS like Rocky Linux or Ubuntu, the cost is $0 (unless you opt for support). If you require enterprise support, RHEL subscriptions cost per node (roughly a few

hundred dollars annually per server). Alternatively, Canonical offers Ubuntu Pro or Advantage support, similarly per node. Since HPC/AI clusters often run without paid OS support (using community distros) to save cost, you may do that and rely on vendor (Dell/Lenovo) hardware support and NVIDIA support for the drivers. Some organizations prefer RHEL for the assured updates – RHEL's license could be ~ $800/node/yr for premium support, which adds up if hundreds of nodes. Rocky or Alma Linux are essentially free RHEL clones and have become popular in HPC after CentOS's changes.

- **Container Orchestration**: Kubernetes itself is free. If you went with an enterprise Kubernetes distribution, costs would appear. For instance, **Red Hat OpenShift** (which is essentially Kubernetes + management tooling + support) is licensed per core. If we consider OpenShift, it might charge, say, ~$2000 per 2 cores for 1 year (just as an example list price). That would be prohibitively expensive at scale (and indeed not necessary if you can do upstream + other tools). The user explicitly said *not to only quote Red Hat*, implying they might not lean towards OpenShift. Another enterprise K8s is **VMware Tanzu** or **Mirantis** – those also cost per node or per cluster. On the other hand, **Rancher (SUSE)** is open-source and free to use; SUSE would charge only if you buy support. So, you could run Rancher's management at no cost and get community support. Given the open-source leaning, likely you'd use **upstream Kubernetes** (with something like Kubespray or Omnia to set it up) and then possibly integrate support through a vendor if needed (some companies get support for Kubernetes from vendors like Canonical or VMware without using their full platform, e.g. Mirantis offers support for pure Kubernetes). Budget-wise, using upstream K8s or Rancher will save a lot compared to OpenShift.

- **HPC Scheduler**: If Slurm is used, it's open source (GNU GPL). You can use it free; however, **SchedMD (the company behind Slurm)** offers enterprise support contracts. They typically price by number of nodes or size of system. For a cluster with, say, a few hundred nodes, a support contract might be on the order of tens of thousands per year. If high availability or custom development is needed, they have tiers. Alternatively, commercial schedulers like **Altair PBS Pro** or **IBM LSF** have per-core or per-socket licensing – those can become very pricey (which is why Slurm dominates HPC now). PBS might charge e.g. a few dollars per core per year – at thousands of cores, that's significant. Since Slurm fulfills the need well and is free, most likely Slurm (with or without a support contract) is the choice for any batch scheduling portion.

- **Run:AI and similar advanced schedulers**: Run:AI (now apparently part of NVIDIA, referenced as **NVIDIA Run:ai**) provides a sophisticated Kubernetes-based GPU scheduling and quota system. It allows setting guaranteed quotas per team, and even **time-slicing of GPUs** for workloads that are okay with a slice of time (important for maximizing utilization) [49] [50]. It also implements **GPU "fractional" sharing** at the software level (so if not using MIG, it can time-slice contexts on a GPU to run more than one container, though isolation is weaker than MIG) [50]. Run:AI was a commercial product (subscription-based, likely per GPU or per cluster pricing). NVIDIA's involvement suggests it might be integrated into NVAIE offerings. Indeed, the NVIDIA software menu shows **"NVIDIA Run:ai"** as part of their stack [51]. This implies that with NVIDIA AI Enterprise, you might get access to Run:AI's scheduling features (which cover multi-tenant quotas, fair scheduling, etc.). If so, that's a big value-add since it directly addresses the multi-tenant fairness challenge. If not included, one would have to license it separately or use open alternatives (there are open-source projects like Volcano/Kube-batch or research schedulers, but they are not as turnkey). For cost planning, if not included with AI Enterprise, something like Run:AI might have cost in the range of a few hundred per

GPU per year (just an estimate, as they never publicly posted prices). Given NVIDIA's push, I suspect they want to include these capabilities in the NVAIE bundle to compete with the cloud experience. So it's worth clarifying with NVIDIA reps whether advanced scheduling (Mission Control or Run:AI) is included or extra.

- **Support and Maintenance**: There's cost in maintaining the cluster software – either internal staff or vendor support. If you partner with Dell or another vendor for a fully managed solution, they might offer a **managed service** or at least comprehensive support (e.g. Dell's managed services or residency services can manage on-prem infrastructure for a fee [52] [53]). With thousands of GPUs and many tenants, you might consider a support contract that covers software stack issues (like NVAIE support calls, driver issues, etc.). NVAIE subscription includes **NVIDIA Business Standard Support** (5-days/week) and you can upgrade to 24/7 **Business Critical** at extra cost [54]. That cost is typically an uplift on the license.

- **Chargeback Model**: For your own revenue model, you'll presumably charge tenants (internal or external) per GPU-hour or via tiers. So metering is not just for cost allocation but for revenue. Tools like LiCO's billing or Kubecost can help calculate usage. For instance, the University of Michigan's Great Lakes cluster publishes rates per GPU-hour (e.g. $x per hour for a V100 GPU) [55]. They even provided a command-line tool for users to estimate cost of a job. In a commercial scenario, you'd decide on pricing that covers the hardware depreciation + software + power/cooling + admin costs + margin. **TCO calculators**: there are a few references that can assist in modeling this. Lenovo published a detailed **Generative AI TCO comparison** showing that for sustained workloads, on-prem can be more cost-efficient than cloud by a large margin [56] [57]. In one example, training a huge model on cloud could cost $480M while on-prem infrastructure to do the same could be far less over its life [58]. Dell's marketing also emphasizes reduced TCO with on-prem/hybrid (Dell APEX, for instance, offers **flexible consumption** so you can pay monthly for on-prem gear, blending CapEx/Opex) [59] [60].

Though not an outright calculator, those whitepapers (Lenovo's, etc.) list cost components: hardware purchase, power usage, cloud instance fees, etc., which you can plug into your scenario. Also, some third-party sites (TotalCAE or cyfuture.cloud) have *rough* cost guides for building GPU clusters [61] [62]. Ultimately, you might create your own spreadsheet model. Key inputs: GPU cost (e.g. an H100 SXM might be ~$30k each, Grace Hopper nodes even more), server cost (with CPUs, memory, etc.), networking (InfiniBand switches can be pricey), storage, facility costs (power distribution, cooling), software licenses (as above), and operations. Spreading that over 3-5 years of depreciation and dividing by expected utilized GPU-hours per year will yield a cost/GPU-hour. Many analyses show if you can keep the cluster fairly busy, the cost per GPU-hour on-prem is much lower than cloud's ~$2-$3 per GPU-hour rates for high-end GPUs [62]. The break-even is usually at high utilization.

For example, if one H100 server (8 GPUs) costs $250k and runs for 5 years, and you manage to utilize it 50% on average, your cost per GPU-hour might be around $0.70 (not including power) – whereas cloud on-demand might be $2-$3/hr for one H100 [63]. This is why many are building "AI factories" in-house now.

There are a few **online TCO calculators** that can help: - **Oracle Cloud vs On-Prem calculator** (Oracle has a generic cloud vs on-prem tool that might be modifiable for GPUs) [64]. - **PhoenixNAP's HPC pricing guide** [65] which explains factors but not an interactive calc. - NVIDIA sometimes provides ROI calculators to partners – not publicly accessible, but an NVIDIA rep might share a spreadsheet for DGX POD vs cloud cost

comparison. - **Lenovo's whitepaper** (LP2225) basically is a detailed TCO study and can serve as a reference for inputs [66] [67].

For **per-node licensing costs** specifically in your design, you'd tally: - Per GPU: $X for NVAIE (unless included). - Per server: OS subscription (if any) + maybe cluster manager if using one per node license. - Per cluster: support contracts for software (Slurm support maybe one contract for cluster, not per node; same for Kubernetes if external support). - If using VMware or something (seems not, but for completeness: VMware vSphere+vGPU licenses would be like $1500 per GPU for vCS license plus vSphere per socket – we likely avoid that here). - If using any database for metadata (like an MLOps metadata store), those might require licenses (e.g. if you used an enterprise DB, probably not though; likely open-source DBs suffice).

Power & Cooling Costs: Not licenses, but need to include in TCO. If each node draws e.g. 3kW and electricity is $0.10/kWh, a node running 24/7 costs ~$2,600/year in power. Multiply by number of nodes. Cooling might add 50% on top. This can actually exceed some software costs, so it's worth noting in budgeting.

In short, the software licenses to build and run a GPU supercluster can be significant but are manageable and often negotiable. By leveraging open-source where possible (Omnia, Kubernetes, Slurm, etc.) you reduce fixed license costs – paying mainly for support (which you can scale as needed). The **big-ticket software** is NVIDIA's AI Enterprise if you choose it; fortunately it's frequently bundled or necessary for supported deployments. Given that your hardware (GPUs, etc.) will be the dominant cost, the software/license costs, even if in the low hundreds of thousands per year for a large cluster, are a smaller fraction of TCO. They are however crucial for providing the multi-tenant features and ease-of-use that differentiate a bare cluster from a full "cloud-like" service.

References:

- NVIDIA, *"NVIDIA AI Enterprise Licensing Guide,"* explains that NVIDIA AI Enterprise is licensed per GPU (one license per GPU in each server) and is available as subscription or perpetual with support [5]. Notably, certain GPUs (H100, H200 NVL, etc.) include AI Enterprise subscriptions for 3-5 years, which can significantly reduce software licensing costs for new hardware [6].

- Dell Technologies, *"Omnia Solution Overview,"* describes the Omnia open-source software stack for deploying converged HPC/AI clusters. Omnia uses Ansible to automate OS provisioning, GPU driver installation, and deployment of workload managers like Slurm and Kubernetes, along with necessary libraries and frameworks [7]. It is released under Apache 2.0 license (fully open source) [9], allowing customization. Omnia enables IT to support mixed workloads (HPC simulation, AI training, data analytics) in one environment with point-and-click templates and profiles, greatly speeding up deployments and reconfigurations [2] [10].

- Lenovo Press, *"On-Premise vs Cloud: Generative AI Total Cost of Ownership,"* provides a TCO analysis for running large-scale AI (LLM training/inference) on-prem vs. public cloud. It notes that sustained, heavy AI workloads tend to be far more cost-efficient on dedicated on-prem infrastructure due to cloud's linear usage costs. For example, training a model on a cloud H100 instance could hypothetically run up hundreds of millions in cloud fees, whereas on-prem hardware (though high upfront cost) would be cheaper over its lifecycle for the same work [58] [56]. The paper breaks down

cost components (CapEx for servers/GPUs vs. OpEx for cloud, power/cooling, etc.) to guide decisions [68] [69].

- Lenovo, *"LiCO 8.0 Administrator Guide,"* outlines features of Lenovo Intelligent Computing Orchestration. LiCO provides a web platform for multi-user AI/HPC clusters, including cluster monitoring, job management, user/group management, and even billing/chargeback functionality. It supports job templates for AI frameworks and HPC jobs, allowing users to submit jobs via web UI easily [41]. It also manages user accounts, quotas and billing groups so that usage can be tracked and charged appropriately [25] [26]. LiCO integrates with common schedulers (Slurm) and supports containerized workloads (with an internal container image repository for AI frameworks) [27] – illustrating a ready-made solution for multi-tenant job submission and resource governance.

- InsideHPC News, *"Supermicro Launches Rack-Scale Systems with NVIDIA Blackwell,"* highlights that Supermicro's turn-key GPU superclusters come with a full suite of integration including **management software**. Their rack-scale designs support up to 768 GPUs across multiple racks, with unified high-performance networking. They natively support NVIDIA's AI Enterprise software for AI workflows [35]. In addition, Supermicro provides **SuperCloud Composer** software to monitor and optimize infrastructure (especially for liquid-cooled systems) [33], delivering a complete solution from hardware to management. This shows hardware vendors bundling software to ease cluster deployment and management at scale.

- NVIDIA Technical Blog, *"NVIDIA Base Command Manager Offers Free Kickstart for AI Cluster Management,"* (June 2025) announces that NVIDIA's Base Command Manager (BCM) is now available at no cost for organizations to deploy on their own clusters [18]. The free license supports clusters of any size (nodes with up to 8 GPUs each) and includes community support. Enterprise support can be purchased if needed, or is bundled with NVIDIA AI Enterprise subscriptions [19]. BCM automates cluster provisioning, workload scheduling, and monitoring for both containerized and non-container workloads [16], enabling multiple users/teams to share GPU clusters securely. This move by NVIDIA provides an "enterprise-grade" cluster management tool without upfront software fees, lowering the barrier for on-prem GPU cloud setups [70].

- NVIDIA, *"Multi-Instance GPU (MIG) Technology,"* explains that MIG can partition a single GPU into as many as **seven independent instances**, each with dedicated resources (SMs, memory, cache) and **full isolation** between them [37]. This allows different workloads or tenants to share one physical GPU with guaranteed Quality of Service. MIG instances can be reconfigured on the fly to adapt to changing demands [39]. By providing hardware-enforced isolation similar to virtual GPUs, MIG enables secure multi-tenancy on GPUs without the overhead of a hypervisor [38]. This is key to maximizing utilization in a multi-tenant GPU cluster while maintaining security/privacy between tenants' workloads.

---

[1] [5] [6] [46] [47] [54] NVIDIA AI Enterprise Licensing — NVIDIA AI Enterprise Licensing Guide
https://docs.nvidia.com/ai-enterprise/planning-resource/licensing-guide/latest/licensing.html

[2] [3] [8] [10] [11] [12] delltechnologies.com
https://www.delltechnologies.com/asset/en-za/products/ready-solutions/technical-support/omnia-solution-overview.pdf

4 NVIDIA AI Enterprise | Cloud-native Software Platform
https://www.nvidia.com/en-us/data-center/products/ai-enterprise/

7 9 GitHub - dell/omnia: An open-source toolkit for deploying and managing high performance clusters for HPC, AI, and data analytics workloads.
https://github.com/dell/omnia

13 Base Command | Operating System of the DGX Data Center - NVIDIA
https://www.nvidia.com/en-us/data-center/base-command/

14 15 16 17 18 19 70 NVIDIA Base Command Manager Offers Free Kickstart for AI Cluster Management | NVIDIA Technical Blog
https://developer.nvidia.com/blog/nvidia-base-command-manager-offers-free-kickstart-for-ai-cluster-management/

20 Accelerate AI on Kubernetes with Spectro Cloud and NVIDIA
https://www.spectrocloud.com/solutions/kubernetes-on-nvidia

21 Virtual Kubernetes clusters with Palette Virtual Clusters
https://www.spectrocloud.com/product/virtual-clusters

22 Release Notes - Spectro Cloud Palette
https://docs.spectrocloud.com/release-notes/

23 24 Palette editions - Superior management, simply packaged - Spectro Cloud
https://www.spectrocloud.com/palette-editions

25 26 27 41 42 48 LiCO 8.0.0 Administrator Guide
https://download.lenovo.com/servers/Lico/EN/lico_8.0_administrator_guide_en_20250408.pdf

28 29 30 31 40 Lenovo Intelligent Computing Orchestration (LiCO) Software Solution | Lenovo US
https://www.lenovo.com/us/en/servers-storage/software/lico/?srsltid=AfmBOooM1sQrg7J1kuv4Tmn3lHJV050LGfiERDScTnjWX0jCJg3cKxbG

32 Lenovo EveryScale HPC & AI Software Stack Product Guide
https://lenovopress.lenovo.com/lp1651-lenovo-hpc-ai-software-stack

33 34 35 36 43 Supermicro Launches Rack-Scale Systems with NVIDIA Blackwell | Inside HPC & AI News
https://insidehpc.com/2025/02/supermicro-launches-rack-scale-systems-with-nvidia-blackwell/

37 39 51 Multi-Instance GPU (MIG) | NVIDIA
https://www.nvidia.com/en-us/technologies/multi-instance-gpu/

38 NVIDIA Multi-Instance GPU User Guide r580 documentation
https://docs.nvidia.com/datacenter/tesla/mig-user-guide/

44 Lambda Builds AI Factories with Supermicro NVIDIA Blackwell GPU ...
https://ir.supermicro.com/news/news-details/2025/Lambda-Builds-AI-Factories-with-Supermicro-NVIDIA-Blackwell-GPU-Server-Clusters-to-Deliver-Production-ready-Next-Gen-AI-Infrastructure-at-Scale/default.aspx

45 NVIDIA AI Enterprise - subscription license (5 years) - 1 GPU - Insight
https://www.insight.com/en_US/shop/product/731-AI7003+P2CMI60/nvidia/731-AI7003+P2CMI60/NVIDIA-AI-Enterprise-subscription-license-5-years-1-GPU/

49 The NVIDIA Run:ai Scheduler: Concepts and Principles
https://run-ai-docs.nvidia.com/self-hosted/2.20/platform-management/runai-scheduler/scheduling/concepts

50 GPU Fractions - NVIDIA Run:ai Documentation
https://run-ai-docs.nvidia.com/self-hosted/platform-management/runai-scheduler/resource-optimization/fractions

52 53 59 60 On-Premise vs. Cloud | Dell USA
https://www.dell.com/en-us/lp/dt/on-premise-vs-cloud

55 Great Lakes HPC Cluster Service Rates
https://its.umich.edu/advanced-research-computing/high-performance-computing/great-lakes/rates

56 57 58 66 67 68 69 On-Premise vs Cloud: Generative AI Total Cost of Ownership > Lenovo Press
https://lenovopress.lenovo.com/lp2225-on-premise-vs-cloud-generative-ai-total-cost-of-ownership

61 HPC Cluster Prices: A Complete Guide | TotalCAE
https://www.totalcae.com/resources/hpc-cluster-price/

62 GPU Cluster Pricing: How Much Does Building One Really Cost?
https://cyfuture.cloud/kb/gpu/gpu-cluster-pricing-how-much-does-building-one-really-cost

63 How Much Can a GPU Cloud Save You? A Cost Breakdown vs On ...
https://www.runpod.io/blog/gpu-cloud-vs-on-prem-cost-savings

64 Cloud Computing Costs in 2024 | Oracle Belize
https://www.oracle.com/bz/cloud/cloud-computing-cost/

65 High Performance Computing (HPC) Price Guide - phoenixNAP
https://phoenixnap.com/blog/hpc-server-price