run:
ai

Run:AI Platform

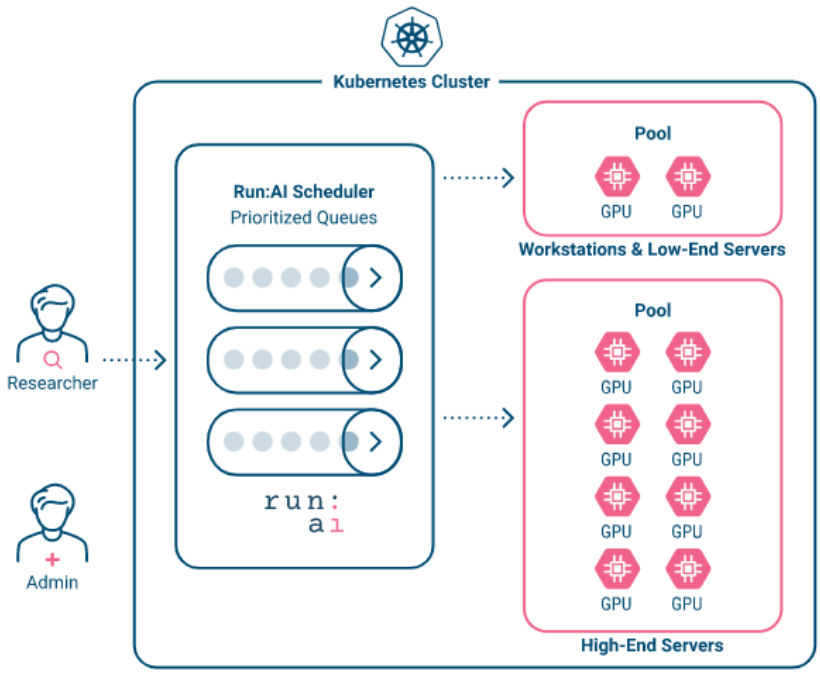
Run:AI abstracts workloads from underlying GPU and compute infrastructure, by creating a shared pool of resources that can be dynamically provisioned, enabling full utilization of GPU compute by various distributed teams within enterprises.

Data science and IT teams gain control and real-time visibility – including seeing and provisioning run-time, queueing, and GPU utilization of each job. In addition to real time visibility the platform also displays historical metrics of cluster resources allowing the enterprise to make more informed, analytical decisions.

A virtual pool of resources enables IT leaders and data scientists to view and allocate compute resources across multiple sites – whether on-premises or in the cloud.

The Run:AI platform is built on top of Kubernetes, enabling simple integration with leading open source frameworks used by data scientists.

Certified by both OpenShift and HPE as a GPU scheduler since no default tools exist in either Orchestration Platforms.



Run:AI Scheduling Capabilities within Kubernetes

Capability	K8s scheduler	Run:AI scheduler for K8s
Scheduling jobs on GPUs	✅	✅
Guaranteed quotas		✅
Automatic queueing/de-queueing		✅
Advanced priorities & policies		✅
Fairness scheduling algorithms		✅
Consolidation & Bin packing		✅
Automatic & dynamic job preemption		✅
Run jobs on GPU fraction		✅
Efficient management of distributed workloads		✅

- **Scheduling jobs on GPUs** - Both schedulers allow the allocation of GPUs to containers orchestrated through Kubernetes. However, Kubernetes requires hard quotas that limit how many GPUs can be used by departments in their respective namespaces. The Run:AI scheduler allows you to virtualize the GPUs into a pool which can be allocated dynamically to the different business units and individuals based on guaranteed quotas rather than just namespaces. This allows teams to get access to more computing power, run more jobs, and essentially be more productive. The usage of the hardware resources becomes more efficient with significantly higher cluster utilization and reduced GPU idle times, thus eliminating the hard quota limit of default Kubernetes.

The following capabilities which are critical for high GPU utilization and business units delivery are unique to Run:AI and are not supported with the default Kubernetes scheduler :

- **Guaranteed quotas**
  With the Run:AI scheduler and guaranteed quotas, the platform ensures that departments at minimum can utilize a defined number of GPU resources. However the scheduler allows departments to also exceed their quota and consume additional idle resources within the cluster greatly increasing GPU utilization. Default Kubernetes scheduling only allows provisioning of resources that are statically assigned to the respective namespace of the department.

- **Automatic queueing/de-queueing**
  The Run:AI scheduler enables data scientists to easily queue many jobs at once that are assigned automatically to available GPU resources based on advanced quotas and priorities,

policies and scheduling algorithms. Once the job has run to completion the workload is detached from the GPU and made available to the next job for scheduling. Automatic queueing/de-queueing allows administrators to take a hands off approach to resource allocation and management while ensuring efficient sharing of resources.

- **Advanced priorities & policies**
  Run:AI includes advanced features that prioritize different job types, define affinity for departments, jobs and GPUs, and ensure termination of idle jobs that are consuming GPU resources. There are a multitude of additional advanced features aimed particularly at data scientists which include job elasticity, hyperparameter optimization, and advanced job reporting.

- **Fairness scheduling algorithms**
  Fairness ensures that GPU resources are proportionally allocated to departments and the respective projects. With default Kubernetes scheduling, resources will have to be statically assigned to different namespaces and departments will only ever be able to consume resources within their namespace. The fairness scheduling algorithms with Run:AI ensure that each project can always utilize their guaranteed resources at any time while also exceeding these quotas to run additional experimentation if idle resources are present. Fairness ensures projects are not being starved by other noisy projects and that resources are always being allocated fairly between the projects.

- **Consolidation & Bin packing**
  The first step in avoiding fragmentation is bin packing where the Run:AI scheduler allocates workloads to currently scheduled nodes before allocating workloads to unutilized nodes.Additionally the scheduler is able to consolidate Jobs on demand. If a workload cannot be allocated due to fragmentation, the scheduler will try to move unattended workloads from node to node in order to get the required amount of GPUs to schedule the pending workload.

- **Automatic & dynamic job preemption**
  The Run:AI scheduler ensures that mission critical workloads are always given highest priority and scheduled onto GPU resources immediately. The scheduler allows the checkpointing and pre-emption of certain workloads to ensure that resources are freed up in order to run higher priority workloads or in order to ensure resources are allocated fairly between projects. In the case of default Kubernetes scheduling, this would require manual intervention by an administrator to find and delete a running workload in order to free up resources for the pending, higher priority workloads.

- **Run jobs on GPU fractions**
  The Run:AI scheduler allows the fractionalization of GPUs within the cluster. This allows multiple workloads to run on a single GPU, increasing job density and experimentation across your resources. With default Kubernetes scheduling, there will always be a one to one mapping between containers and GPUs regardless of the underlying GPU resources that are required and sufficient to run the workload.

- **Efficient management of distributed workloads**
  The Run:AI scheduler supports distributed workloads that run across multiple nodes to orchestrate large scale training jobs. It efficiently schedules these jobs to ensure efficient and fair utilization of GPU resources within the cluster. Kubernetes can run distributed workloads however it often causes starvation and fragmentation of the underlying resources.