backend.AI | lablup

Feature Highlight

Backend.AI Slurm Frontend

Keep the Way You Work, Replace What Is Behind It

Backend.AI Slurm Frontend lets an HPC site move its workloads onto Backend.AI without changing how its Slurm users work. It takes the place of the Slurm controller and accounting service, so researchers keep using the Slurm commands and scripts that they already know while every job runs as an isolated container on Backend.AI.

The cost of moving off Slurm

Many HPC and AI teams want what container platforms offer, such as isolated software environments, GPU sharing and a clear view of who is using what. Very few can simply switch. Their job scripts, workflow engines and accounting reports depend on Slurm commands behaving exactly as they always have, and every one of those dependencies has to be found, rewritten and revalidated before the old platform can be retired.

Replacing the Slurm servers with Backend.AI

Slurm Frontend implements Slurm's network protocol and returns results in Slurm's formats, but runs no Slurm scheduler or compute daemons. Each request becomes the equivalent Backend.AI operation. Features without a Backend.AI equivalent are refused up front with a Slurm error instead of running with different semantics.

Why Slurm Frontend?

Nothing to relearn
Existing job scripts, wrappers and pipelines keep working.

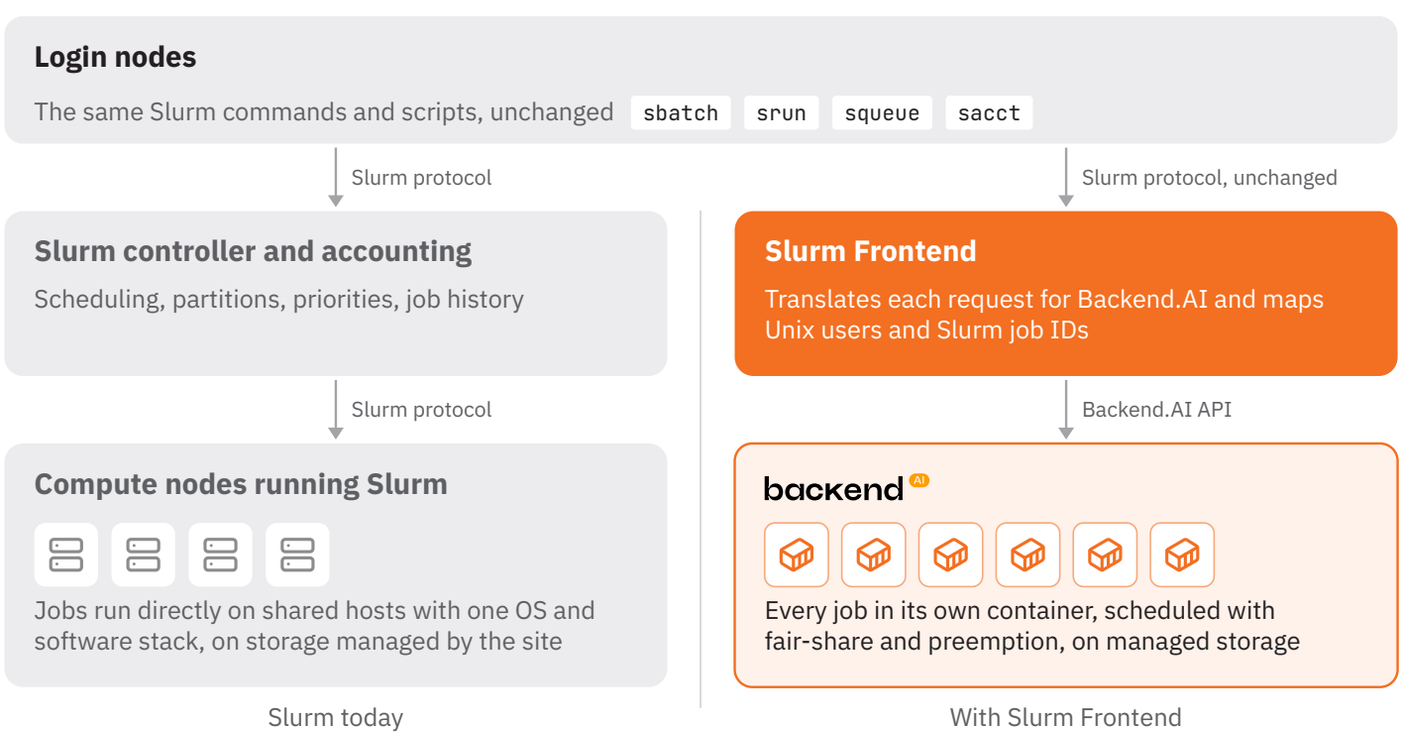
A container per job
Users choose an image per job, so software stacks never collide.

Migration without the rewrite
The interface stays, so pipeline migration largely disappears.

Usage you can see
Per-user and per-project usage and quotas in a web console.

One platform for all workloads
Batch, interactive and AI workloads share the same nodes and storage.

Running Slurm jobs on Backend.AI with existing commands and scripts



Running jobs with Slurm commands

Backend.AI handles each everyday Slurm command.

Command	On Backend.AI
`sbatch`	Batch submission, including job arrays, dependencies, deferred starts and held jobs
`salloc`	Interactive allocations
`srun`	Interactive shells and commands, including a real shell in the job's container through `srun --pty bash`

`squeue`
`sinfo`
`scontrol`
	Queue and cluster views

`scancel`	Cancellation
`sacct`	Job history, kept after the underlying sessions are cleaned up
`sshare`	Fair-share data computed by Backend.AI's fair-share scheduler

When a partition has priority tiers, Backend.AI's scheduler can stop lower-priority jobs on the same nodes to make room for a waiting high-priority job, and Slurm tools show those jobs as preempted.

Mapping Slurm concepts to Backend.AI

Most* Slurm concepts map one-to-one onto Backend.AI.

Slurm concept	Becomes in Backend.AI
Batch job (`sbatch`)	Batch session that runs the job script
Interactive job (`salloc`, `srun`)	Interactive session
Job step (`srun` inside a job)	A command run inside the job's container
Multi-node job	Multi-node session spanning several containers
Partition	Resource group, optionally paired with a project
GPU request (GRES)	Accelerator resource slots
Unix user	Backend.AI user with the same Unix identity
Shared filesystem	Managed storage folders mounted into each job
Software environment	Container image

* Slurm features without a Backend.AI equivalent are caught at submission, so every job that runs behaves as its script expects.

Isolation and hardware access for each job

Each job runs in its own container, from an image the user chooses, under the submitting user's Unix identity. That identity maps to a Backend.AI account, so jobs are counted against the right project and are subject to its access rules and quotas. Scratch and dataset folders and a persistent home directory can be mounted into every job from managed storage.

Slurm GPU requests are served from Backend.AI's GPU pool. On NVIDIA GPUs, Container-level GPU Virtualization lets Slurm jobs and other Backend.AI workloads share a device in fractions. Backend.AI schedules AI accelerators from 12 vendors, including NVIDIA, AMD, Intel, Google, Graphcore, Rebellions, FuriosaAI, HyperAccel and Tenstorrent. Partitions map onto pools of these accelerators, so the same `sbatch` and `srun` commands work on whichever hardware runs the job.

Inside the container, MPI and NCCL workloads use the same hardware paths they would on bare metal. NUMA-aware placement keeps a job's CPU cores and accelerators close together, and the node's InfiniBand and RoCE adapters are reachable directly, with pinned-memory limits lifted for RDMA. Multi-node jobs exchange data GPU to GPU with GPUDirect RDMA, and GPUDirect Storage streams data from supported storage into GPU memory.

Supported environment

Backend.AI	26.8.1 or later, with every Slurm Frontend feature including preemption
Slurm clients	Standard clients from several releases; 24.11 recommended
Authentication	Munge, using the site's existing setup
Container runtime	Docker or Podman

About Backend.AI

Backend.AI is an open-source operating system built for AI and HPC clusters. It pools a site's GPUs and other AI accelerators so users get isolated environments on demand, while administrators manage projects, quotas and policies in one place. It runs at more than 120 sites, on premises, in the cloud or air-gapped.

Get started

Reach out to your Lablup partner or contact Lablup at contact@lablup.com to bring Slurm Frontend to your Backend.AI cluster.

backend.AI | lablup

© 2026 Lablup Inc. All rights reserved. Oct 22 Slurm is a registered trademark of SchedMD LLC.
All other trademarks are the property of their respective owners.

github.com/lablup/backend.ai

www.backend.ai