Welcome to the AIRCC Cluster

A Researcher's Guide to Israel's AI Resource Compute Center

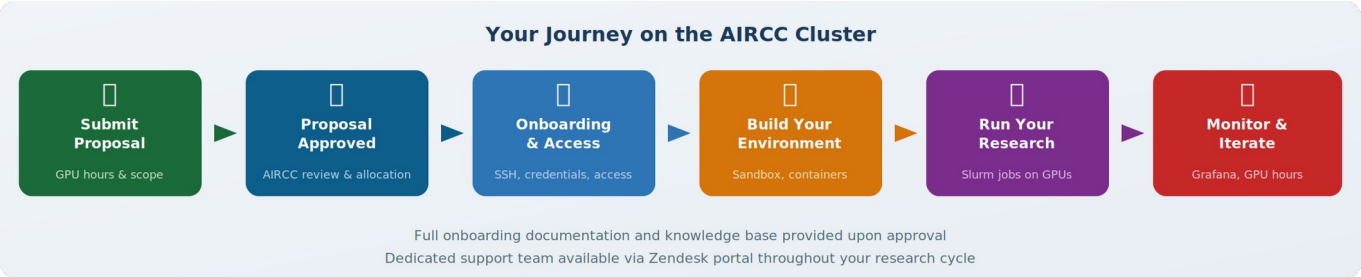
For research teams considering a proposal submission

The AI Research Compute Center is a national-scale GPU cluster built to give Israeli researchers the compute power they need for cutting-edge AI, deep learning, and scientific computing work. It is managed and operated by **TZAG Elita Compute Solutions** and serves research groups from all state funded universities and colleges.

This guide is designed to give you — a researcher considering submitting a proposal — a clear, honest picture of the platform: what the environment looks like, what technical skills are expected, and what practical considerations you should think about as you plan your work. This is *not* an onboarding manual; full documentation is provided once your proposal is accepted.

From Proposal to Results

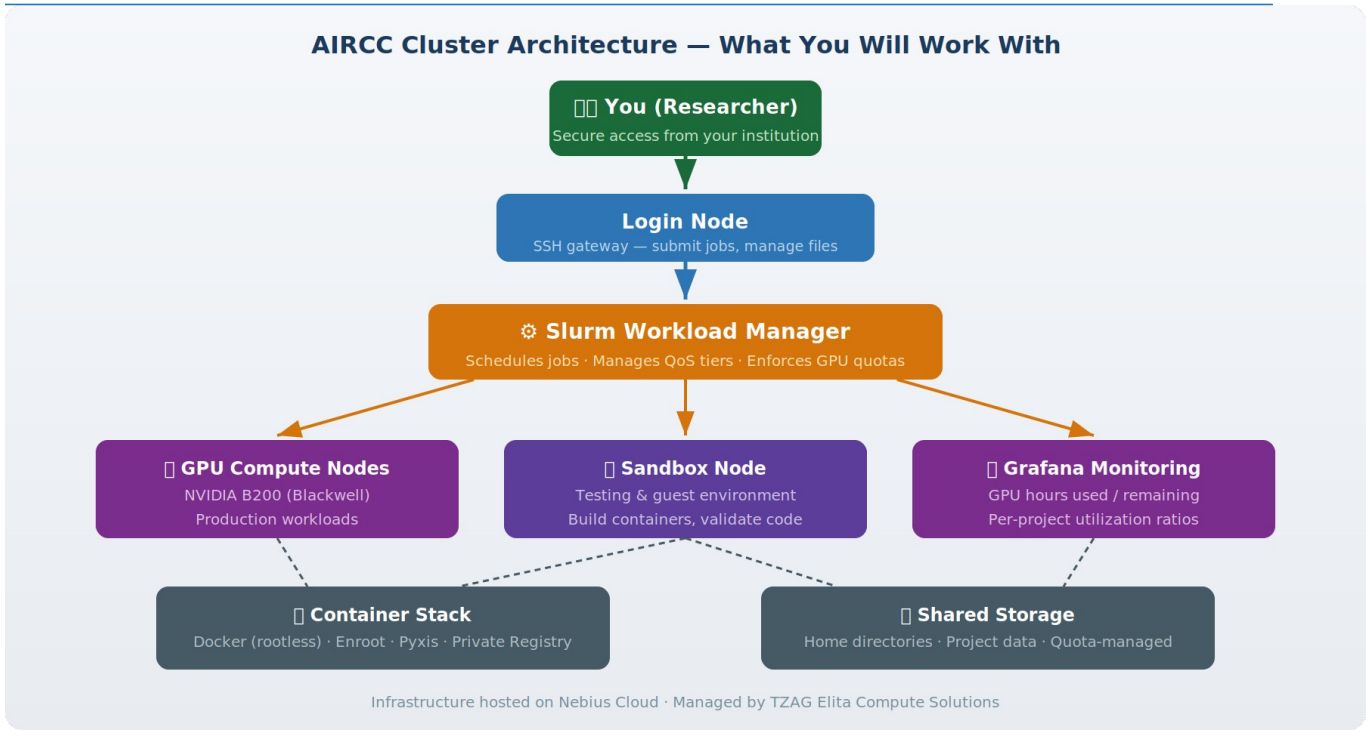
The path from writing your proposal to running production experiments follows a clear, supported workflow. Here is the big picture:



You don't need to be an expert in every layer of this stack before you apply. **You do need to be comfortable working in a Linux terminal, writing and submitting batch jobs, and containerizing your code.** If you have trained models on a few GPUs and are ready to scale up, this platform is built for you.

The Computing Environment

The AIRCC cluster is a production HPC environment hosted on Nebius Cloud infrastructure. The diagram below outlines the main components you will interact with:



Key Specifications

GPU Fleet	NVIDIA B200 (Blackwell architecture) GPU compute nodes
Workload Manager	Slurm — industry-standard HPC job scheduler with QoS-based prioritization
Container Stack	Docker (rootless), Enroot, Pyxis — build in a sandbox, push to a private registry, run via Slurm
Sandbox	A dedicated testing and guest environment for building containers, validating code, and preparing workloads before submitting to the production compute partition
Access Method	Access from institutional networks + SSH key-based authentication
Storage	Shared filesystem with per-user and per-project quotas
Monitoring	Grafana dashboards for GPU hours tracking, utilization ratios, and budget status
Support	Dedicated support team via Zendesk portal, comprehensive FAQ, and an extensive knowledge base with step-by-step guides

What Technical Skills Will I Need?

The AIRCC cluster is a real research HPC environment — not a managed notebook service. That said, the bar is achievable for any PhD student with some hands-on AI/ML programming experience. Here is a realistic breakdown:

Essential (You Should Have This)

- **Linux command line:** Comfortable navigating directories, editing files, managing processes via SSH. This is your primary interface.
- **Python + deep learning frameworks:** PyTorch or TensorFlow at a working level. You should be able to write a training loop, not just call a high-level API.
- **Basic Docker:** You'll build containers in the sandbox environment. If you've written a Dockerfile and built an image before, you have what you need.
- **GPU-aware training:** Experience running training on at least one GPU. Understanding of CUDA at a user level (device placement, memory management).

Helpful (You'll Learn as You Go)

- **Slurm basics:** sbatch, srun, squeue — how to submit and monitor jobs. Detailed documentation is provided in the knowledge base.
- **Multi-GPU / distributed training:** NCCL, DistributedDataParallel, data parallelism. Valuable for large-scale experiments but not required for every project.
- **Container registries:** Pushing/pulling images from a private registry. The workflow is guided in the onboarding docs.

Not Required

- You do not need to be a system administrator, network engineer, or Slurm expert. The support team handles infrastructure. Your job is to focus on your research.

Planning Your Proposal — Practical Considerations

Before you write your proposal, it's worth thinking through some practical aspects of running research on a shared HPC cluster. These aren't hurdles — they're things that, with a little planning, make the difference between a smooth experience and a frustrating one.

Data: Getting It In, Keeping It Safe

- **The AIRCC cluster is not a data storage solution.** All research data must be maintained in your own institutional or external repository. Data copied to the cluster is solely for the purpose of efficient computation — the shared filesystem is a working area, not an archive.
- **Plan your data pipeline before you start.** Storage is quota-managed. Consider both your input dataset sizes and your expected output sizes (model checkpoints, logs, generated results) — these all count toward your quota and should be part of your proposal thinking.
- **External network access is restricted.** You cannot download datasets from the internet during a running job. Pre-stage your data on the shared filesystem before submitting.
- **Use public repositories where possible.** Datasets from sources like HuggingFace Hub, Kaggle, or institutional data stores should be downloaded and staged during your sandbox preparation time, not at job runtime.
- **After offboarding, prompt data download is required.** When your allocation cycle ends, you are expected to download your data and results promptly. Data remaining on the cluster after the offboarding period may be removed.

GPU Hours & Utilization: Use Them Wisely

- **Your allocation is measured in GPU hours.** When you request 8 GPUs for 24 hours, that's 192 GPU hours from your budget. Track this via the Grafana dashboard.
- **Idle GPUs still count.** If your job allocates 4 GPUs but only uses 2, you're spending 4 GPUs worth of time. Profile your workload and right-size your requests.
- **Start small, then scale.** Use the sandbox to validate your code on a small run before requesting a full-scale job on the compute partition.

Checkpointing: Your Safety Net

- **Always implement checkpointing.** This is non-negotiable for long-running training jobs. Save model state periodically so you can resume if a job is interrupted.
- **Jobs can be preempted.** On a shared cluster with QoS-based scheduling, higher-priority jobs may preempt lower-priority ones. Your job will receive a SIGTERM signal with a 15-minute grace period — use it to save state cleanly.
- **Design for restartability.** Write your training script to detect an existing checkpoint and resume from it automatically. This is best practice on any HPC system.

Job Design: Think Before You Submit

- **Estimate your runtime.** Slurm needs a time limit for every job. Overestimate slightly, but don't request 72 hours when your job takes 6 — it blocks resources.

- **Containerize your full environment.** The cluster uses Pyxis and Enroot to run your containers natively within Slurm. Build your image in the sandbox, push it to the private registry, and reference it in your batch script. This ensures reproducibility.
- **Log everything.** Direct stdout/stderr to files. When something fails at 3 AM, your logs are all you have.

Getting the Most Out of Your Allocation

GPU hours are a shared, finite resource. A few practical habits can help you use your allocation more effectively and avoid unnecessary delays.

Start Early in the Cycle

- **Don't wait until the last weeks of the allocation cycle to begin.** Early in the cycle the cluster is typically less loaded, which means shorter queue times and more flexibility to experiment. Researchers who start early have more room to iterate on their approach before the end-of-cycle rush.

Understand Why Jobs Are Pending or Rejected

- **Check the Grafana dashboard.** If your jobs are sitting in the queue or being rejected, the dashboard will show you why — whether it's cluster load, quota limits, or resource constraints. Understanding the current state of the cluster helps you adjust your requests accordingly.
- **Open a support ticket if you're stuck.** If the dashboard doesn't make the reason clear, reach out via the Zendesk portal. The support team can help you diagnose scheduling issues and suggest alternatives.

Right-Size Your Jobs

- **Consider splitting large jobs.** A single job requesting many GPUs for many hours may wait longer in the queue. Think about whether your workload can be broken into smaller parallel jobs that each need fewer GPUs — or the same number of GPUs but for shorter durations. Smaller requests often get scheduled faster.
- **Match GPUs to actual need.** If your training script only uses 2 GPUs effectively, don't request 8. Over-requesting wastes your allocation and delays other researchers.

Monitor CPU and GPU Utilization

- **Check your GPU utilization in Grafana.** Low GPU utilization during a job usually means your data pipeline is the bottleneck — the GPUs are waiting for data. Consider increasing the number of DataLoader workers, enabling pin_memory, or pre-processing your data into a faster-loading format.
- **Watch your CPU utilization too.** If CPU usage is low while GPUs are also underperforming, you may be under-utilizing available CPU cores for data preprocessing. Adjusting num_workers in your DataLoader or parallelizing data augmentation can make a significant difference.

- **Use profiling tools.** PyTorch Profiler, NVIDIA Nsight Systems, or even simple timing logs can reveal where your training loop spends its time. A few minutes of profiling can save hours of wasted GPU time.

Support & Knowledge Base

The AIRCC platform comes with a dedicated Zendesk support portal staffed by the TZAG Elita team. Beyond ticket-based support, researchers have access to a comprehensive FAQ covering the most common questions, and an extensive knowledge base with step-by-step guides to help you at every stage of your work. Available guides include:

- Accessing Grafana — Cluster Monitoring
- Onboarding Session — Cycle 1 Recording
- Default Cluster Containers
- PyTorch DataLoader Recommendations
- How to Login to the Login Node
- Copy Your Data
- Cluster Commands
- Preemption Checkpointing Guide
- Slurm Build Workflow
- Institutional Technical Support

These resources are available through the Zendesk portal upon onboarding and are continuously updated as the platform evolves.

Ready to Apply?

The AIRCC cluster was built for researchers like you. Whether you're training large language models, running computational biology simulations, or pushing the boundaries of computer vision — this platform gives you access to state-of-the-art GPU compute with professional support and infrastructure.

You don't need to know everything before you begin. What matters is a solid research plan, a willingness to work in a Linux/HPC environment, and a realistic understanding of the resources you need. Our support team and documentation are here to bridge the gaps.

> **Questions about the proposal process or the platform?**
>
> Reach out through your institution's AIRCC coordinator or contact the TZAG Elita support team. We're here to help you plan a successful research project.