Run:AI Call Preparation Guide

# ✅ Pre-Call Checklist

* Understand Your Current Setup:  
   - Cluster setup: OCP version, node types (e.g. H200 GPUs), namespaces.  
   - Existing GPU scheduling method (if any).  
   - Your use case: Model training, inference, multi-tenant environment?
* Access Details:  
   - Do you already have Run:AI installed on your cluster?  
   - Do you have access to the dashboard/UI?  
   - Are you using Run:AI CLI or SDK?
* Prepare Your Questions:  
   - How does Run:AI handle fractional GPU allocation?  
   - Can you isolate resources per team/user/project?  
   - How is scheduling different from Kubernetes native GPU support?  
   - How does Run:AI integrate with OpenShift (OCP)?  
   - What metrics and logging are available?  
   - How are GPU quotas and limits set?

# 📌 Key Topics You Might Expect Them to Present

* Run:AI Architecture Overview:  
   - How Run:AI scheduler works on top of Kubernetes.  
   - Integration with GPU vendors (NVIDIA).  
   - Core components (scheduler, controller, CLI, UI).
* Features Overview:  
   - Fractional GPU support.  
   - Prioritized queuing and resource management.  
   - Experiment tracking and user isolation.  
   - Integration with JupyterHub and CI/CD pipelines.
* Multi-Tenancy and Quota Management:  
   - How teams or projects are isolated.  
   - Configuring GPU pools, time-sharing, and fairness policies.
* Monitoring & Logging:  
   - Built-in dashboards.  
   - Integration with Prometheus/Grafana or ELK.
* Licensing and Support:  
   - Open-source vs. enterprise differences.  
   - Support model and typical onboarding time.

# 🗣️ Sample Questions to Ask

* How do you recommend isolating GPU usage between dev, test, and prod namespaces?
* Can Run:AI help us schedule specific GPU types (e.g., H200 vs V100)?
* How is scheduling optimized when multiple users submit jobs at the same time?
* How does Run:AI interact with OpenShift security constraints or SCCs?
* What are best practices for tracking experiments across teams?