GPU Validation Checklist

# Hardware Availability

* GPU is physically present or attached (e.g., check with `nvidia-smi`)
* Correct number of GPUs are recognized
* GPU model matches expected (e.g., H200, A100, V100, etc.)

# Driver and CUDA Compatibility

* NVIDIA driver is installed and compatible with the GPU model
* CUDA toolkit is installed and matches driver version
* `nvidia-smi` shows healthy status and no errors

# Resource Accessibility

* User has permissions to access GPU (check container/namespace/user role)
* GPU is not being overutilized or locked by another process
* Run a small GPU job to check compute access (e.g., simple matrix multiplication)

# GPU Utilities and Monitoring

* `nvidia-smi` shows usage stats correctly (temperature, memory, processes)
* NVML (NVIDIA Management Library) accessible via Python or CLI
* Monitoring tools like DCGM, Prometheus, Run:AI, or MIG manager (if applicable) are running

# Software Environment

* PyTorch / TensorFlow detects GPU (`torch.cuda.is\_available()` or `tf.config.list\_physical\_devices('GPU')`)
* Conda/venv includes proper GPU packages (e.g., `torch` with CUDA backend)
* Any required libraries for inference/training are installed (e.g., `transformers`, `onnxruntime-gpu`)

# Performance & Stress Test (Optional but Good)

* Run `nvidia-smi dmon` or `stress-ng` to test sustained load
* Benchmark tool runs (e.g., `mlperf`, `resnet50`, `bert`)

# Fractionalization/Virtualization (if using Run:AI, Kubernetes, etc.)

* Validate MIG/Run:AI fractional GPU setup if applicable
* Each pod/container gets the expected GPU resources
* Isolation works between different GPU workloads

# Storage and Data Pipeline

* GPU nodes can read/write to storage (NAS/S3/PVC etc.)
* Data loading doesn’t bottleneck the GPU compute