# LLM Validation Checklist for OCP AI + Run:AI Setup

## ✅ Goal

Validate the deployment of Large Language Models (LLMs) on OpenShift AI using Run:AI and H200 GPUs, integrated with CI/CD tools such as Harness, GitHub, Anaconda, and Artifactory.

## 🔹 PHASE 1: Environment & Infrastructure Validation

* ✅ Namespace Ready  
  - Description: Ensure the OCP AI namespace is correctly created and has required access  
  - How to Validate: oc get project / login test  
  - Tools: OpenShift CLI
* ✅ GPU Nodes Accessible  
  - Description: GPU nodes (e.g. H200) are visible and schedulable  
  - How to Validate: oc get nodes -l nvidia.com/gpu.present=true  
  - Tools: OCP CLI
* ✅ Run:AI Operator Installed  
  - Description: Operator up and running in target namespace  
  - How to Validate: oc get pods -n runai  
  - Tools: OpenShift
* ✅ PVC / S3 Mounted  
  - Description: Data access available  
  - How to Validate: Test ls /mnt/pvc or s3fs mount inside pod  
  - Tools: PVC / S3 logs
* ✅ Internet or Artifactory Access  
  - Description: Models/code can be pulled  
  - How to Validate: Try cloning Git repo or pulling Docker image  
  - Tools: Git / Artifactory
* ✅ Correct Base Image / Conda Env  
  - Description: Required Python/CUDA/PyTorch present  
  - How to Validate: conda list or nvcc --version  
  - Tools: Anaconda / Shell
* ✅ Harness Pipeline Trigger  
  - Description: Pipeline triggers on Git push or manual  
  - How to Validate: Test build + deploy steps  
  - Tools: Harness CI/CD

## 🔹 PHASE 2: LLM Deployment Validation

* ✅ Model is Loaded Successfully  
  - Description: Model loads in memory without crash  
  - How to Validate: Logs show “model loaded” or first inference  
  - Tools: main.py logs
* ✅ GPU is Utilized  
  - Description: Model runs on H200 GPU  
  - How to Validate: nvidia-smi / torch.cuda.is\_available()  
  - Tools: Pod shell
* ✅ Tokenizer and Config Load  
  - Description: Tokenizer + config are correctly loaded from repo  
  - How to Validate: No JSON/dependency errors  
  - Tools: HuggingFace or custom loader
* ✅ Model Precision  
  - Description: Correct precision (fp16, bf16, int8, etc.) set  
  - How to Validate: Log/debug output or model.half()  
  - Tools: PyTorch debug
* ✅ Batch Size Supported  
  - Description: Test batch input (for RAG / chatbot)  
  - How to Validate: No OOM, handles input size  
  - Tools: Inference script
* ✅ Prompt + Response Working  
  - Description: Output is valid for sample prompt  
  - How to Validate: Try: “What is AI?” → sensible answer  
  - Tools: Script/UI
* ✅ Post-processing Steps Work  
  - Description: Trimming, decoding, etc. works  
  - How to Validate: Response format is readable  
  - Tools: Python logs/UI
* ✅ Validation Tests Pass  
  - Description: Any unit tests, smoke tests, or Harness jobs succeed  
  - How to Validate: pytest, unittest  
  - Tools: Harness, VSCode

## 🔹 PHASE 3: Performance, Monitoring & UI

* ✅ Inference Time Logged  
  - Description: Log duration from prompt to response  
  - How to Validate: Use time.perf\_counter() or similar  
  - Tools: Python logs
* ✅ GPU Utilization Visible  
  - Description: In Run:AI or OCP metrics dashboard  
  - How to Validate: Check GPU memory use, utilization %  
  - Tools: Run:AI UI
* ✅ Streamlit / Gradio UI Loads  
  - Description: UI is reachable, interactive  
  - How to Validate: Access via port-forward / ingress  
  - Tools: Browser / Logs
* ✅ Logs Stored for Debug  
  - Description: stdout/stderr are saved  
  - How to Validate: Check pod logs  
  - Tools: oc logs <pod>
* ✅ Job Completion Status  
  - Description: Jobs finish successfully  
  - How to Validate: runai describe job or Harness status  
  - Tools: Run:AI / Harness

## 🔹 BONUS: Model-Specific Validation

🧠 Chat Models (GPT, LLaMA, Mistral): Prompt completeness, coherent response, safe output

📄 Summarization Models: Factual summary, no hallucination

📚 RAG Pipelines: Retrieval is relevant, generated answer grounded

🧪 Fine-Tuned LLMs: Output matches use case (e.g., finance, legal, etc.)