INTERVIEW PREP · GPU INFRASTRUCTURE

The GPU infrastructure interview, taken apart.

What an Oracle Principal Core Infrastructure loop actually tests, the six troubleshooting questions that decide it, and the reasoning path a principal engineer is expected to walk out loud.

> Nobody passes this loop by knowing more acronyms. You pass it by narrowing a problem under pressure, in front of someone who will keep asking why.

1. Where the loop actually spends its time

Write down every phone-screen question you remember, especially the ones you could not answer cleanly. Loop questions are often similar or identical to the screen. That one exercise is worth more than any amount of fresh reading.

AREA	SHARE	WHAT GOOD PREPARATION LOOKS LIKE
GPU architecture and troubleshooting	25%	H100/H200/B200, NVLink/NVSwitch, PCIe, NUMA, Xid/ECC, DCGM, node validation
Kubernetes and Slurm	20%	GPU scheduling, device plugin, node states, pending jobs, GRES, drain and return-to-service
Networking and RDMA	15%	RoCE/InfiniBand, GPUDirect RDMA, topology, congestion, drops, MTU, NIC health
Linux and node troubleshooting	15%	CPU, memory, OOM, I/O wait, systemd, dmesg, cgroups, NUMA
Resume deep dives	15%	Scope, architecture, metrics, tradeoffs, failures, and your personal contribution
Behavioural and OCI values	10%	Ownership, ambiguity, conflict, failure, influence, learning, measurable outcomes

2. The shape of every good answer

Six steps, in this order, whatever the domain. Say them out loud as you go. An interviewer who can hear the structure stops testing whether you have one.

01 Clarify scope and symptoms.

02 State the most likely failure domains.

03 Start with the highest-signal, lowest-cost checks.

04 Use telemetry or commands to prove or disprove each hypothesis.

05 Mitigate blast radius if production is affected.

06 Find root cause, validate the fix, and explain prevention.

3. The six domains

For each one: what you need to know, the question you will actually be asked, and the order in which you are expected to eliminate things. The path is the part worth

memorising. It is not an answer, it is a route.

GPU NODE

Know. H100 vs H200 vs B200; SXM vs PCIe; HBM; NVLink/NVSwitch; MIG; ECC/Xid; drivers and CUDA; DCGM; PCIe/NUMA topology.

> One GPU in an 8-GPU node is 40% slower. Walk me through troubleshooting.

Good path. Application → utilisation and memory → thermals and clocks → ECC/Xid → NVLink → PCIe/NUMA → network → hardware.

CLUSTER AND DISTRIBUTED

Know. Single-node vs multi-node bottlenecks; NCCL; stragglers; topology; scheduler placement; network and storage coupling.

> Training is fast on one node but much slower on 32 nodes. Why?

Good path. GPU compute → NCCL → NVLink → NIC/RDMA → fabric congestion → topology → stragglers → storage and data pipeline.

SLURM

Know. slurmctld/slurmd, partitions, GRES, node states, draining, job states, scheduling, health checks.

> A GPU job is stuck Pending. How do you determine why?

Good path. Reason field → requested resources → partition constraints → GRES → node state → reservations and limits → scheduler logs.

KUBERNETES

Know. NVIDIA device plugin and GPU Operator, allocatable GPU resources, kubelet, taints and tolerations, affinity, runtime and CNI.

> A pod requests a GPU but stays Pending even though GPUs appear available.

Good path. Describe pod → scheduler reason → allocatable GPU → device plugin → node health → taints and affinity → fragmentation and driver state.

NETWORK

Know. TCP/IP, DNS, MTU, bonding, RDMA, RoCE/InfiniBand, GPUDirect RDMA, congestion, packet drops, topology.

> Nodes communicate successfully, but distributed training performance is poor.

Good path. Separate connectivity from performance: throughput and latency → errors and drops → RDMA counters → MTU → congestion → topology and oversubscription.

LINUX AND STORAGE

Know. CPU load, memory and OOM, I/O wait, NUMA, kernel logs, filesystem; IOPS vs throughput, metadata, NVMe cache, parallel storage.

> GPU utilisation is only 30%. What do you inspect?

Good path. CPU and data loader → memory and NUMA → storage → network → synchronisation → application pipeline, before blaming the GPU.

4. The resume deep dive

Any strong bullet on your resume is a ten-minute technical interrogation waiting to happen. Be precise about ownership, and never hide behind "we".

> What was the exact scale: nodes, GPUs, workloads, users?

> What was the architecture, and why was it designed that way?

> What problem or bottleneck were you solving?

> What metrics proved both the problem and the improvement?

> What did you personally design, implement, decide or troubleshoot?

> What alternatives did you consider, and what tradeoffs did you make?

> What failed or surprised you?

> What would you do differently today?

If you claim H100 or B200 clusters, Grafana node validation, Slurm failures or inference optimisation, be ready to explain the actual telemetry, the topology, the root cause and the before-and-after number.

5. The three behavioural questions

Q1. Tell me about a time you had a major production issue or infrastructure failure. How did you handle it?

> **They are testing.** Ownership under pressure, troubleshooting discipline, communication, prioritisation and prevention.

> **Answer shape.** Situation → impact → what you did → hypotheses and evidence → mitigation → root cause → permanent fix → measurable result → learning.

> **Expect follow-ups.** What was the root cause? What did you do first? How did you know? What would you do differently? How did you prevent recurrence?

> Avoid "the team investigated and fixed it". Show your personal decisions and your technical reasoning.

Q2. Tell me about a time you had to make a technical decision without all the information you needed.

> **They are testing.** Judgment under ambiguity, risk management, learning speed, and knowing when to escalate.

> **Answer shape.** Unknowns → constraints → separate knowns from assumptions → identify the highest-risk assumption → proof of concept → rollback criteria → decision → result → learning.

> **Expect follow-ups.** What assumption was wrong? What if the decision had failed? Who disagreed? What evidence changed your mind?

> The principal signal here is distinguishing reversible from irreversible decisions, and showing how you reduced uncertainty before committing.

Q3. Tell me about a technical decision you made that turned out to be wrong.

> **They are testing.** Self-awareness, accountability, adaptability and learning.

> **Answer shape.** Decision → why it looked reasonable then → evidence you were wrong → how quickly you changed course → impact → what you changed in your decision process.

> **Expect follow-ups.** Pick a real mistake with meaningful consequences, not a disguised strength.

> Avoid blaming incomplete requirements, another team or a junior engineer. Own the decision and show the improvement in judgment.

6. STAR, compressed

SITUATION	TASK	ACTION	RESULT
Set context and stakes quickly.	What were you accountable for?	Spend most of your time here. Use "I". Explain why.	Quantify the impact, and say what you learned.

7. Sixty seconds before you join

> Do I remember every phone-screen question, and my weak answers?

> Can I explain an 8-GPU node and a multi-node training path end to end?

> Can I troubleshoot a bad node, a Pending Slurm job, and a Pending GPU pod?

> Can I explain RDMA/RoCE/InfiniBand at a practical troubleshooting level?

> Can I defend every GPU, Slurm, Grafana and Kubernetes claim on my resume?

> Do I have six to eight STAR stories with measurable outcomes?

> Am I asking clarifying questions before jumping to a solution?

> Am I saying what I personally did, not only what the team did?

Do not try to sound encyclopedic. Sound systematic. A principal engineer makes the interviewer confident that when the answer is not obvious, they know how to narrow the problem, use evidence, manage risk, and drive it to closure.

Where to go from here

This sheet tells you what you will be asked. It cannot make you rehearse, and the part engineers leave until last is never the GPU question.

WORKBOOK

The Behavioural Interview Story Bank

Section 7 asks whether you have six to eight STAR stories ready. This is how you start: five worksheets that map eight to ten candidates against the nine areas, then develop your strongest one in full.

- Five worksheets, about 45 minutes start to finish
- A coverage matrix that shows you the areas you have no story for yet
- A pressure test that finds the unsourced number before an interviewer does
- Prints in black and white, with room to write

Get the workbookhttps://askgurpreet.com/ebooks/workbook-story-bank

$4.99 one payment

Also available with **Mastering the Interview**, 112 answers written out in full.

The Interview Bundle https://askgurpreet.com/ebooks/bundle-interview **$12.99** ~~$14.98~~