PassQueen
a.k.a dumpsqueen.com

NVIDIA AI Infrastructure

NVIDIA NCP-AII

Version Demo

Total Demo Questions: 16

Total Premium Questions: 163

Buy Premium PDF

https://passqueen.com

support@passqueen.com

Topic Break Down

Topic	No. of Questions
Topic 1, AI Infrastructure Design and Planning	20
Topic 2, AI Infrastructure Deployment	55
Topic 3, AI Infrastructure Operations	30
Topic 4, AI Infrastructure Troubleshooting and Optimization	58
Total	163

QUESTION NO: 1

After configuring NGC CLI with ngc config set, a user receives Authentication failed errors when

pulling containers. What step was most likely omitted?

A. Installing the CLI with apt-get instead of manual extraction.

B. Entering the API key during ngc config set or storing it in ~/.ngc/config.

C. Setting --format_type=json to enable API interactions.

D. Running sudo systemctl restart docker after configuration.

E. Running `docker login nvcr.io` with `$oauthtoken` as the username and the NGC API key as the password.

ANSWER: E

Explanation:

Running `ngc config set` configures the NGC CLI, but pulling a container image through Docker requires Docker itself to authenticate to the NVIDIA Container Registry at `nvcr.io`. The omitted step is logging Docker in to that registry with the NGC API key. NVIDIA documents this flow as `docker login nvcr.io`, using `$oauthtoken` as the username and the generated NGC API key as the password. Docker then saves the registry credential in its Docker configuration and can present it when it requests protected image manifests and layers.

This distinction is important because the CLI configuration file and Docker's registry credential store are separate. A valid API key may be supplied to `ngc config set` for NGC CLI operations, while Docker still has no registry login session unless the Docker login command has been performed. The API key must belong to an account authorized for the relevant NGC organization or container, and it should be handled as a secret. NVIDIA's container registry instructions describe generating an API key and using it to authenticate Docker before executing a `docker pull`: NVIDIA NGC User Guide. CLI configuration behavior is also covered in the NGC CLI User Guide.

QUESTION NO: 2

You are tasked with setting up High Availability (HA) for NVIDIA Base Command Manager (BCM) in a new GPU cluster. The cluster consists of a primary head node, a secondary head node, and several compute nodes. The requirements are automatic failover of BCM services, minimal disruption to workloads, and proper cluster health monitoring during and after installation. During your BCM HA installation and configuration process, which two of the following actions are mandatory for ensuring a robust and verified HA cluster configuration?

Pick the 2 correct responses below.

A. Assign a floating Virtual IP address that can automatically migrate between the primary and secondary head nodes during failover.

B. Compute nodes must be powered on and performing work to initiate synchronization of the head nodes.

C. After configuration is complete, simulate a failover by stopping BCM services on the active head node to verify that all services are running on the secondary node with no interruption.

D. Configure both head nodes to use independent static IP addresses for BCM services instead of relying on a shared virtual IP address.

E. During configuration, explicitly synchronize both the configuration and state data directories from the primary to the secondary head node to ensure consistency.

ANSWER: A C

Explanation:

Assign a floating Virtual IP address that can automatically migrate between the primary and secondary head nodes during failover is required because it gives users, compute nodes, and external management integrations one stable endpoint for BCM services. Rather than requiring every client to determine which head node is currently active, HA moves the service-facing virtual address to the surviving head node. This is fundamental to transparent service access following a head-node failure or planned switchover.

After configuration is complete, simulate a failover by stopping BCM services on the active head node to verify that all services are running on the secondary node with no interruption is also essential as a validation activity. HA configuration is not considered operationally verified merely because the initial setup commands succeed. A controlled failover confirms that the standby node can assume the active role, the virtual address migrates as expected, and BCM service health can be checked from both head nodes. This test should be performed during an approved maintenance or validation window, so that service transition behavior and workload impact can be observed safely. NVIDIA's BCM documentation and DGX BasePOD BCM reference material describe HA operation and validation around active/standby head nodes and service continuity. See NVIDIA Base Command Manager Documentation and NVIDIA DGX BasePOD BCM Reference Architecture.

QUESTION NO: 3

A cluster administrator is creating a Slurm job submission script for an eight-GPU distributed training job. Which two settings help ensure that the job receives the requested GPU resources and does not start until all requested resources are available? (Pick the 2 correct responses below)

A. Specify `#SBATCH --gpus=8` in the job script.

B. Specify `#SBATCH --wait-all-nodes=1` for coordinated multi-node startup.

C. Set `CUDA_VISIBLE_DEVICES=all` globally on every cluster node.

D. Run the job from the Slurm controller without requesting an allocation.

E. Disable cgroup enforcement for all job steps.

ANSWER: A B

Explanation:

The `--gpus=8` directive requests eight GPUs for the job allocation. Slurm uses configured generic resources to select nodes that can satisfy this request.

The `--wait-all-nodes=1` directive instructs Slurm to wait until all allocated nodes are ready before beginning the job step. This is useful for coordinated distributed training because all ranks must begin with the expected resources available. GPU visibility inside each task should still be validated through the scheduler integration and application launch configuration. Slurm documents GPU resource requests and allocation behavior in the Slurm Generic Resource documentation.

QUESTION NO: 4

A user needs to configure NGC CLI to access resources across multiple organizations. What is the recommended command syntax to achieve this?

A. export NGC_CLI_ORG=org-name & & ngc config set

B. ngc config list to manually edit the JSON configuration file.

C. ngc registry login --org org-name

D. ngc config set --org org-name --ace ace-name

ANSWER: D

Explanation:

The recommended command is `ngc config set --org org-name --ace ace-name`. The NGC CLI stores a current configuration context, including the NGC organization and the associated API key/credential context (ACE). Setting both values explicitly lets an administrator or user switch the active CLI context to the organization whose private registry, containers, models, or other NGC resources they need to use. This is particularly important when the same user has access to more than one organization, because organization-scoped resources and permissions are evaluated against the configured organization context. The command can be run again with the appropriate organization and ACE values whenever the user needs to work with another organization. NVIDIA documents the CLI configuration workflow and the `ngc config set` command as the way to establish these settings, rather than editing CLI configuration files directly. See the NGC CLI installation and configuration documentation and the NGC CLI command reference for configuration details.

QUESTION NO: 5

A platform team needs to collect GPU utilization, memory, temperature, power, and Xid-related telemetry from every Kubernetes GPU worker node into Prometheus. Which two actions are required? (Pick the 2 correct responses below)

A. Deploy DCGM Exporter as a DaemonSet on GPU worker nodes.

B. Configure Prometheus to scrape the DCGM Exporter metrics endpoint.

C. Install the CUDA Toolkit in every application pod.

D. Configure GPU persistence mode as a replacement for metrics collection.

E. Run a separate NVIDIA driver version for every monitored workload namespace.

ANSWER: A B

Explanation:

Deploying `dcgm-exporter` as a DaemonSet ensures that a metrics-exporter pod runs on each eligible GPU worker node. DCGM Exporter obtains GPU telemetry through NVIDIA Data Center GPU Manager and exposes the selected metrics in Prometheus format.

Prometheus must also be configured to scrape the exporter endpoint, either directly or through a Prometheus Operator resource such as a ServiceMonitor. Without a scrape configuration, the exporter can expose metrics locally but those metrics are not collected into the monitoring system. NVIDIA documents exporter deployment and metric collection in the DCGM Exporter documentation.

QUESTION NO: 6

An administrator needs to verify HA functionality after configuring BCM (Bright Cluster Manager). Which command confirms the active head node and failover readiness?

A. cmsh status to check HA status and active/standby roles.

B. nvsm show health to validate GPU status on both head nodes.

C. systemctl restart cmdaemon to force a failover test.

D. ping < secondary-head-node-ip > to test basic connectivity.

E. cmha status to display the active head node, standby node, and HA service status.

ANSWER: E

Explanation:

`cmha status` is the appropriate Bright Cluster Manager command for validating the state of a configured high-availability head-node pair. The `cmha` utility manages and reports BCM HA services, so its status output identifies the node serving as the active head node and the peer's standby state. This provides an operational view of whether the HA configuration is

healthy enough to support a controlled transition of the cluster-management role if the active node becomes unavailable. It is therefore the relevant verification after HA configuration, because the check is made against the HA subsystem rather than against only an individual host, hardware component, or network path. Administrators should run the command from a head node with the necessary BCM administrative privileges and confirm that the expected active/standby roles and HA services are reported as healthy before considering the deployment failover-ready. NVIDIA's Base Command Manager documentation is available through the NVIDIA Base Command Manager Documentation portal; NVIDIA also maintains Bright Cluster Manager product and support resources at the NVIDIA Bright Cluster Manager page. These resources document the platform's head-node management and HA operational model.

QUESTION NO: 7

After a firmware upgrade on a DGX H100, the administrator notices that one GPU is not detected by the system. Which troubleshooting step should be performed first to identify the root cause?

A. Review firmware update logs and run nvsm show health to check for hardware or firmware errors on the affected GPU.

B. Remove the GPU from the system and replace it with a new one before any diagnostics.

C. Ignore the issue and proceed with production workloads if the other GPUs are operational.

D. Immediately re-run the firmware upgrade on all system components.

ANSWER: A

Explanation:

Review firmware update logs and run `nvsm show health` to check for hardware or firmware errors on the affected GPU. This is the appropriate first action because it gathers evidence before making a potentially disruptive change. The firmware-update output and logs establish whether the GPU-related firmware components completed successfully and whether an activation requirement, such as a reboot or power cycle, remains outstanding. NVSM is NVIDIA's DGX system-management utility and provides a platform-level view of system health, including GPU status and associated hardware telemetry. Its health information can help distinguish a firmware activation or inventory issue from a condition that requires deeper PCIe, GPU tray, BMC, or hardware investigation. Administrators should also validate the GPU's visibility with the normal GPU inventory tools after collecting this initial diagnostic data, then follow the documented remediation indicated by the findings. This evidence-based sequence preserves useful failure details and supports an appropriate escalation if a component-level fault is identified. NVIDIA documents NVSM monitoring and health capabilities in the NVIDIA System Management User Guide. DGX H100 platform firmware procedures and applicable update guidance are available through the NVIDIA DGX H100 User Guide documentation.

QUESTION NO: 8

What is the primary purpose of performing a NeMo burn-in on a new AI infrastructure?

A. To benchmark production training speed and ensure all GPUs are running at identical clock speeds.

B. To stress test the hardware and software stack with representative NeMo workloads, ensuring reliability.

C. To tune NeMo model hyperparameters for maximum accuracy on user datasets during cluster deployment.

ANSWER: B

Explanation:

To stress test the hardware and software stack with representative NeMo workloads, ensuring reliability, is correct because a burn-in validates that the complete AI platform operates dependably under realistic distributed-training conditions before production use. Unlike a basic GPU-discovery check or an isolated microbenchmark, a NeMo workload exercises the integrated path used by AI training: GPU compute and memory, CUDA software, multi-node GPU communication through NCCL, networking, storage throughput for data and checkpoints, and the container and scheduling environment. Running this workload for an appropriate duration helps expose intermittent infrastructure problems, such as communication instability, GPU or driver errors, thermal or power-related behavior, and I/O limitations, that may not appear during short

functional tests. The outcome is confidence that the system can sustain representative AI workloads reliably, along with actionable evidence for investigating any failures or performance instability found during validation. NVIDIA's NeMo Framework is designed for scalable generative-AI training and customization workloads, making it suitable for representative end-to-end validation. NVIDIA also documents NCCL as the library that enables collective communication across GPUs and nodes, a critical component that workload burn-in exercises in distributed environments. See the NVIDIA NeMo Framework overview and the NCCL documentation.

QUESTION NO: 9

When configuring an out-of-core HPL burn-in for a 40B matrix on 8x H100 nodes, which environment variable prevents GPU out-of-memory errors while reserving space for drivers?

A. export HPL_OOC_SAFE_SIZE=4.0

B. export HPL_OOC_MODE=0

C. export HPL_OOC_NUM_STREAMS=8

D. export HPL_OOC_MAX_GPU_MEM=90

ANSWER: A

Explanation:

`export HPL_OOC_SAFE_SIZE=4.0` is correct because `HPL_OOC_SAFE_SIZE` controls the amount of GPU memory, measured in GiB, that NVIDIA HPL out-of-core mode deliberately leaves unused. This reserved capacity is available for the NVIDIA driver, CUDA runtime, communication buffers, and other GPU-memory overhead that is not part of the HPL matrix allocation. Reserving 4 GiB is a practical safeguard during a large burn-in workload, where an allocation that appears to fit nominal H100 memory can otherwise fail because usable memory is lower than the physical framebuffer capacity. Out-of-core HPL can place portions of the matrix in system memory while staging data to the GPUs, but it still requires enough free device memory for execution and supporting runtime allocations. Setting this variable therefore reduces the chance that a 40B-matrix run terminates with an out-of-memory condition caused by insufficient driver/runtime headroom, rather than by a hardware issue. NVIDIA documents `HPL_OOC_SAFE_SIZE` as the GPU-memory safety margin for HPL out-of-core operation; its HPL guidance also recommends increasing the safe size when out-of-memory errors occur. See the NVIDIA HPL Benchmark documentation and the NVIDIA HPC Benchmarks documentation.

QUESTION NO: 10

After a DGX system GPU tray maintenance procedure, an engineer needs to validate that all eight GPUs are visible in the expected topology and that intra-node collective communication is functioning. Which two actions are appropriate? (Pick the 2 correct responses below)

A. Run `nvidia-smi topo -m` to inspect the GPU topology matrix.

B. Run `all_reduce_perf -b 8 -e 16G -f 2 -g 8` to exercise all local GPUs.

C. Run `ibstat` to validate the NVLink Switch fabric.

D. Enable GPU persistence mode as a topology validation test.

E. Run a one-GPU CUDA application to validate eight-GPU collectives.

ANSWER: A B

Explanation:

Running `nvidia-smi topo -m` displays the GPU topology matrix and relevant interconnect relationships. This allows the engineer to confirm that GPUs are visible and that the reported topology is consistent with the expected platform design after maintenance.

Running `all_reduce_perf -b 8 -e 16G -f 2 -g 8` executes an NCCL all-reduce performance test across all eight local GPUs. The test exercises collective communication over the intra-node GPU interconnect and provides bandwidth results that can be compared with the platform baseline. NVIDIA documents topology reporting in the NVIDIA System Management Interface documentation and collective testing concepts in the NVIDIA nccl-tests repository.

QUESTION NO: 11

A System Administrator needs to change the scheduling behavior of a single GPU to use a fixed share scheduler. What command achieves this?

A. esxcli system module parameters set -m nvidia -p

B. esxcli -i 0 -mig 18

C. nvidia-smi -i 0 -mig 1

D. mlxconfig -d /dev/mst/mt4123_pciconf0 set LINK_TYPE_P1 =2

E. nvidia-smi -i 0 -sched 1

ANSWER: E

Explanation:

The command `nvidia-smi -i 0 -sched 1` sets the compute scheduling mode of GPU index 0 to fixed-share scheduling. The `-i 0` selector limits the change to the individual GPU identified by index 0, rather than applying a setting across all GPUs in the host. The `-sched` (compute scheduling) setting controls how GPU execution time is scheduled in environments that support these scheduler modes. A value of `1` selects fixed-share scheduling, which assigns each vGPU a fixed share of the GPU's processing time. This is appropriate when an administrator needs predictable allocation behavior for vGPU workloads on a specific physical GPU. NVIDIA documents the GPU scheduling-mode controls as part of its vGPU management workflow and identifies fixed share as a supported scheduling policy. The command is executed on the host with sufficient administrative privileges and should be validated against the installed NVIDIA vGPU software version and the GPU's supported features. See the NVIDIA vGPU User Guide and the NVIDIA System Management Interface documentation for the scheduling-mode syntax and platform requirements.

QUESTION NO: 12

Refer to the output:

~ $ sudo nvsm show healthinfo

"Timestamp: Sat Dec 16 16:26:32 2017 -0800

Version: 17.12-5

Checks"BIOS Revision [5.11]..... DGX Serial Number [YSY72800016).....

Verify installed DIMM memory sticks. Healthy

...[output truncated)

Verify Ethernet controllers. Healthy

Verify installed GPU's Unhealthy

Checking output of 'lspci' for expected GPU's Missing GPU at PCI address '07:00.0'

Verify installed InfiniBand controllers. Healthy

Verify PCIe switches. Healthy

...[output truncated)

What insights can a system administrator gain regarding the DGX system's health?

A. A GPU tray upgrade failed.

B. A GPU is missing on the DGX system.

C. A GPU driver upgrade has failed.

D. The system has passed the hardware health check successfully.

ANSWER: B

Explanation:

A GPU is missing on the DGX system. The NVSM health report explicitly marks the installed-GPU verification as `Unhealthy` and identifies a missing expected device at PCI address `07:00.0`. The check uses PCIe enumeration information, including output from `lspci`, to verify that the DGX hardware currently visible to the operating system matches the expected platform GPU inventory. Therefore, the administrator can conclude that one GPU is not being detected by the system at the expected PCIe location.

This result is actionable as a hardware-visibility condition: the administrator should investigate the GPU and its associated PCIe connectivity and platform hardware state. Useful follow-up checks include confirming the PCIe inventory with `lspci`, reviewing GPU visibility with `nvidia-smi` if the driver is loaded, and collecting NVSM health or support information before servicing the system. NVIDIA documents `nvidia-smi` as the utility for reporting and managing NVIDIA GPU devices, while DGX documentation provides the platform-specific operational and support workflow context. See the NVIDIA System Management Interface documentation and the NVIDIA DGX documentation portal.

QUESTION NO: 13

Which of the following steps are essential components of a recommended DGX cluster installation procedure?

Pick the 2 correct responses below.

A. Group nodes by function during initial setup and assign them to relevant categories in the cluster management tool.

B. Configure networking by validating all interfaces on each node, ensuring proper InfiniBand and Ethernet connectivity prior to installing cluster software.

C. Install Slurm on the head node and then configure the compute nodes' default OS images.

D. Complete application containerization, run distributed jobs, and skip validation of node health or storage availability.

ANSWER: A B

Explanation:

Group nodes by function during initial setup and assign them to relevant categories in the cluster management tool is an essential installation practice because DGX cluster deployment relies on clear roles and repeatable configuration. Separating management, login, compute, storage, and other infrastructure functions enables the cluster-management platform to apply the appropriate software image, services, policies, and operational procedures consistently to each group. This structure also supports subsequent lifecycle operations, such as provisioning, monitoring, updates, and scaling. NVIDIA's Base Command Manager documentation describes device categories as a core mechanism for organizing and managing cluster nodes and associated infrastructure.

Configure networking by validating all interfaces on each node, ensuring proper InfiniBand and Ethernet connectivity prior to installing cluster software is equally essential. DGX clusters depend on correctly functioning management Ethernet, high-speed InfiniBand or Ethernet fabrics, and—where applicable—storage networking. Connectivity validation before software-stack deployment establishes that cabling, addressing, routing, fabric configuration, and interface status are sound. It provides the reliable infrastructure required for node provisioning, distributed training communication, scheduler operation, and shared-storage access. NVIDIA's DGX SuperPOD deployment guidance emphasizes completing and validating the physical and network infrastructure as part of the deployment process. See NVIDIA Base Command Manager documentation and NVIDIA DGX SuperPOD documentation.

QUESTION NO: 14

You are an infrastructure engineer tasked with validating a new AI training cluster before releasing it to users. Your team wants to perform a NeMo burn-in to ensure both hardware and software are reliable and ready for production workloads. Which of the following actions are required as part of a proper NeMo burn-in process?

Pick the 2 correct responses below.

A. Download a pre-trained NeMo model and use it for a quick accuracy check on a user dataset, then consider the burn-in complete if results are reasonable.

B. Test inference using the NeMo API and approve the environment if the model outputs valid predictions.

C. Run the configured NeMo training job repeatedly or for an extended duration, monitoring for errors, stalls, or performance drops across all GPUs and nodes.

D. Configure a representative NeMo training or pretraining recipe and set up an executor to launch the job across intended nodes and GPUs.

ANSWER: C D

Explanation:

A NeMo burn-in must exercise the cluster with a realistic distributed training workload rather than merely confirming that an inference request succeeds. Configuring a representative NeMo training or pretraining recipe and an executor is required so that the workload is launched over the intended GPU and node topology. This validates the operational path used in production, including the framework configuration, container/runtime environment, scheduler or execution backend, GPU allocation, and distributed-process setup. NVIDIA NeMo Framework recipes provide the configuration structure for training and pretraining workloads, while NeMo Run executors define where and how those workloads are launched.

The configured job must then run repeatedly or for a sustained period while the team monitors all participating GPUs and nodes for failures and degradation. Extended execution places meaningful load on GPU compute and memory, host resources, interconnect communication, storage, and the distributed-training software stack. It helps identify intermittent faults such as GPU errors, collective-communication hangs, node instability, and throughput regressions that may not appear during a short functional test. This combination establishes both launch correctness and reliable operation under production-like training pressure. See the NVIDIA NeMo Run execution guide and NVIDIA NeMo training documentation.

QUESTION NO: 15

A platform team is enabling Multi-Instance GPU (MIG) on supported GPUs for Kubernetes workloads. Which two statements are correct? (Pick the 2 correct responses below)

A. MIG can partition a supported GPU into isolated compute and memory instances.

B. Kubernetes must discover the configured MIG resources before they can be scheduled.

C. MIG combines the memory of multiple physical GPUs into one address space.

D. MIG eliminates the need for an NVIDIA driver on the host.

E. Every NVIDIA GPU supports MIG mode.

ANSWER: A B

Explanation:

MIG partitions a supported physical GPU into isolated GPU instances with dedicated compute and memory resources. This allows a scheduler to allocate a MIG device to a workload rather than assigning the full physical GPU.

After the desired MIG configuration is applied, the NVIDIA device plugin and GPU Operator inventory must reflect the configured instances so Kubernetes can advertise and schedule the resulting MIG resources. MIG is not a method for

combining memory from separate physical GPUs into one larger memory pool. NVIDIA documents MIG operation and deployment considerations in the NVIDIA Multi-Instance GPU User Guide.

QUESTION NO: 16

An administrator is preparing a Kubernetes cluster for GPU workloads using the NVIDIA GPU Operator. Which two prerequisites should be validated on every intended GPU worker node? (Pick the 2 correct responses below)

A. A supported NVIDIA GPU is installed in the worker node.

B. The worker node runs a supported Linux distribution and kernel.

C. Every Kubernetes control-plane node has an NVIDIA GPU installed.

D. The kubelet is configured to disable extended resources.

E. Docker is the only supported container runtime.

ANSWER: A B

Explanation:

A supported NVIDIA GPU and a compatible Linux kernel environment are required because the GPU Operator deploys and manages components that interact with the GPU hardware and, when configured to do so, installs kernel-facing software such as the NVIDIA GPU driver. The node must also be supported by the selected operator and driver release.

Container runtime support is also required. GPU-enabled containers depend on a configured runtime, such as containerd, that can use the NVIDIA Container Toolkit integration to expose the GPU devices and driver capabilities to workloads. A control-plane node does not need a GPU merely because GPU workloads will be scheduled elsewhere. See the NVIDIA GPU Operator getting started documentation and the NVIDIA Container Toolkit documentation.