NCP-AIOオンライン試験 & NCP-AIO日本語版テキスト内容

P.S. GoShikenがGoogle Driveで共有している無料かつ新しいNCP-AIOダンプ：https://drive.google.com/open?id=1-QfFsRz0olbp0pDpnzdOPxuNqwTqTK49

NCP-AIO認定の取得を支援するために、多くの専門家が数年間、NVIDIAすべての試験官向けのNCP-AIO試験トレントを策定するために懸命に取り組んできました。GoShiken このようにして、当社のNCP-AIO学習資料は、対象となるだけでなく、すべての知識ポイントを網羅しています。 NCP-AIO練習教材には、NCP-AIO練習教材の学習プロセスの欠陥を見つけるのに役立つ統計分析機能もあるため、弱いリンクのNVIDIA AI Operationsトレーニングを強化できます。 このようにして、能力が向上したため、成功に自信を持つことができます。

NVIDIA NCP-AIO 認定試験の出題範囲：

トピック	出題範囲
トピック 1	• トラブルシューティングと最適化：NVIこの試験セクションでは、AIインフラストラクチャエンジニアのスキルを評価し、高度なAIシステムで発生する技術的な問題の診断と解決に焦点を当てます。出題範囲には、Docker、NVIDIA NVlinkおよびNVSwitchシステムのFabric Managerサービス、Base Command Manager、Magnum IOコンポーネントのトラブルシューティングが含まれます。受験者は、ストレージパフォーマンスの問題を特定して解決し、AIワークロード全体で最適なパフォーマンスを確保する能力も実証する必要があります。

トピック 2	• 管理：このセクションでは、システム管理者のスキルを評価し、データセンターにおけるAIワークロードの管理に不可欠なタスクを網羅します。受験者は、フリートコマンド、Slurmクラスタ管理、そしてAI環境に特有のデータセンターアーキテクチャ全体を理解していなければなりません。また、Base Command Manager（BCM）、クラスタプロビジョニング、Run.ai管理、そしてAIとハイパフォーマンスコンピューティングアプリケーションの両方に対応したマルチインスタンスGPU（MIG）の構成に関する知識も求められます。
トピック 3	• ワークロード管理：このセクションでは、AIインフラストラクチャエンジニアのスキルを評価し、AI環境におけるワークロードの効率的な管理に焦点を当てます。Kubernetesクラスターの管理、ワークロード効率の維持、システム管理ツールを用いた運用上の問題のトラブルシューティング能力を評価します。NVIDIAテクノロジーと連携し、異なる環境間でワークロードがスムーズに実行されることを重視します。
トピック 4	• インストールと展開：このセクションでは、システム管理者のスキルを測定し、インフラストラクチャのインストールと展開に関するコアプラクティスを扱います。受験者は、Base Command Manager のインストールと設定、NVIDIA ホストでの Kubernetes の初期化、NVIDIA NGC およびクラウド VMI コンテナーからのコンテナーの展開についてテストされます。また、AI データセンターにおけるストレージ要件の理解、DPU Arm プロセッサーへの DOCA サービスの展開、AI ドリブン環境の堅牢な構築についても学習します。

権威のあるNCP-AIOオンライン試験 & 資格試験におけるリーダーオファー & 信頼できるNVIDIA NVIDIA AI Operations

我々はNVIDIAのNCP-AIO試験問題と解答また試験シミュレータを最初に提供し始めたとき、私達が評判を取ることを夢にも思わなかった。我々が今行っている保証は私たちが信じられないほどのフォームです。NVIDIAのNCP-AIO試験はGoShikenの保証を検証することができ、１００パーセントの合格率に達することができます。

NVIDIA AI Operations 認定 NCP-AIO 試験問題 (Q57-Q62):

質問 # 57
You need to implement a highly available and fault-tolerant Fleet Command deployment for a mission-critical AI application. What architectural considerations are MOST important for ensuring resilience?

- A. Rely solely on the edge devices' ability to operate independently during Fleet Command outages.
- B. Deploy a single Fleet Command server in a standard configuration.
- C. Monitor the Fleet Command server's performance and resource utilization.
- D. Regularly back up the Fleet Command server's configuration and data.
- E. Utilize a multi-node Fleet Command cluster with redundancy and failover mechanisms, combined with geographically diverse edge device deployments.

正解：E

解説：
A multi-node cluster provides redundancy and failover capabilities, ensuring high availability. Geographically diverse edge deployments minimize the impact of regional outages. A single server (A) is a single point of failure. Backups (C) are important but don't prevent downtime. Monitoring (D) helps identify issues but doesn't ensure resilience. Solely relying on edge devices (E) limits manageability and control.

質問 # 58
You need to monitor the GPU utilization of pods in your Kubernetes cluster running AI workloads. You want to use Prometheus to collect these metrics. Which of the following approaches is most suitable and efficient for collecting GPU metrics within the Kubernetes environment?

- A. Use the Kubernetes metrics server to collect GPU utilization metrics directly from the kubelet on each node.
- B. Install the 'gpu-exporter' as a Deployment. Configure Prometheus to scrape these endpoints.
- C. Run 'nvidia-smi' inside each pod and expose the output as a Prometheus metric using a custom script and the Prometheus client library.
- D. Manually SSH into each node and run 'nvidia-smi' to record GPU utilization, then input the data into Prometheus.
- E. Install the 'nvidia-dcgm-exporter' as a DaemonSet to expose GPU metrics from each node. Configure Prometheus to scrape these endpoints.

正解：E

解説：
The correct answer is A. The 'nvidia-dcgm-exporter' is designed to efficiently expose GPU metrics from each node as Prometheus endpoints. Running it as a DaemonSet ensures that it runs on every node with GPUs. Option B is inefficient and requires significant overhead. Option C, the Kubernetes metrics server, does not collect detailed GPU utilization metrics. Option D is a manual and impractical approach. Option E is not standard and relies on an unspecified 'gpu-exporter' , whereas 'nvidia-dcgm-exporter' is an official solution.

質問 # 59
The 'nvsm' service is consistently crashing on one of your nodes. Analyzing the core dump reveals a segmentation fault related to memory access within the NVSwitch driver. What is the MOST appropriate course of action?

- A. Increase the swap space on the node.
- B. Report the issue to NVIDIA support with the core dump and relevant system information.
- C. Disable the NVSwitch on the affected node.
- D. Try different versions of CUDA.
- E. Recompile the NVSwitch driver with debugging symbols.

正解：B

解説：
Segmentation faults related to driver code usually indicate a bug within the driver itself. Reporting the issue with a core dump allows NVIDIA engineers to investigate and provide a fix. Trying to debug the driver yourself (recompiling) or disabling the NVSwitch are less effective solutions for this type of problem. Different versions of CUDA also can cause problems, but first report with the core dump.

質問 # 60
Your application, which relies heavily on NVLink for inter-GPU communication, is experiencing performance degradation over time. After investigating, you suspect that NVLink link errors are accumulating. How can you proactively monitor NVLink link error counts and trigger an alert when they exceed a predefined threshold? (Select TWO correct answers)

- A. Analyze the system's kernel log for NVLink-related error messages.
- B. Implement a custom script that periodically reboots the GPUs to clear the error counters.
- C. Use 'nvsm show links' and parse the output to extract error counts, then integrate this into a monitoring system.
- D. Configure 'nvsm' to automatically restart the NVLink connections when errors are detected.
- E. Use 'nvidia-smi' to query NVLink error counters and integrate the output into a monitoring system (e.g., Prometheus, Grafana).

正解：C、E

解説：
'nvsm show linkS (or a similar 'nvsrn' command) and 'nvidia-smr are both capable of providing NVLink error counts. The key is to then integrate the output of these commands into a monitoring system that can trigger alerts based on predefined thresholds. 'nvsm' doesn't have native auto-restart features for links based on errors. Periodically rebooting GPUs is a poor workaround. Kernel logs can provide some information, but it is not an effective way of real time monitoring.

質問 # 61
You are tuning the storage performance of a Kubernetes cluster that uses Longhorn as the storage backend. You've noticed inconsistent read latencies. What Longhorn specific configurations could you adjust to potentially improve the consistency and

reduce latency for read operations?

- A. Increase the number of replicas for each Longhorn volume to improve data redundancy and read parallelism.
- B. Reduce the size of the Longhorn volumes to decrease the amount of data that needs to be read.
- C. Enable 'data locality' to ensure that replicas are preferably placed on the same node as the pod accessing the volume, reducing network latency.
- D. Disable Longhorn's built-in monitoring to reduce overhead.
- E. Adjust the 'replica-auto-balance' setting to redistribute replicas more evenly across nodes.

正解：A、C、E

解説：
More replicas increase read parallelism. 'replica-auto-balance' ensures replicas are well-distributed. 'data locality minimizes network hops for reads. It is not beneficial to reduce volume size, as that is required. Disabling built-in monitoring is also a bad idea.

質問 # 62
......