Task 03: Kubernetes in Details

Creating a NVIDIA GPU Kubernetes POD and Node Pool

NVIDIA GPU Setup on GKE:

1. **Create a GKE Cluster:**

- Use the Google Cloud CLI (gcloud) to create a GKE cluster with the specified configuration, including Ubuntu containerd node images and NVIDIA GPU accelerators.
- Refer to the GKE documentation for detailed instructions.

NVIDIA GPU Setup on Azure AKS:

1. **Install NVIDIA Device Plugin:**
- Use a DaemonSet to deploy the NVIDIA device plugin on each node in the cluster, providing necessary drivers for the GPUs.
- Follow the steps in the Azure AKS GPU Workload documentation for manual installation.

2. **Configure GPU Node Pool**:
- Create a GPU-enabled node pool in your AKS cluster. Specify the GPU type and count per node in the node pool configuration.

Dealing with Memory Leaks in NVIDIA GPU Kubernetes POD:

1. **Monitor Resource Usage:**
- Use Kubernetes monitoring tools (e.g., Prometheus, Grafana) to monitor resource usage metrics such as memory consumption in the NVIDIA GPU pod.

2. **Identify Memory Leak:**
- Monitor the memory usage trend over time. If you observe a continuous increase in memory consumption without release, it indicates a memory leak.

3. **Troubleshooting:**
- Use debugging tools like kubectl exec to access the pod and investigate the application's memory usage. Check application logs for any error messages related to memory allocation or leaks.

Calculating Resource Consumption of the Application

1. **Analyze Application Requirements:**

- Understand the application's resource requirements, including CPU, GPU, memory, and storage. Consult application documentation or perform benchmarking tests to determine resource usage patterns.

2. **Kubernetes Resource Requests and Limits:**

- Specify resource requests and limits in the pod specification YAML file based on the application's resource consumption metrics. This ensures that Kubernetes allocates sufficient resources to the pod and enforces resource constraints.

3. **Dynamic Resource Allocation:**

- Implement dynamic resource allocation with Kubernetes Horizontal Pod Autoscaler (HPA) and Vertical Pod Autoscaler (VPA) to scale resources based on demand.
- This allows Kubernetes to adjust resource allocation automatically in response to workload changes