Contents

Monitoring HPE GreenLake Servers running GPU using Grafana and Prometheus

Overview

HPE GreenLake provides a cloud-native platform for managing and monitoring infrastructure with built-in tools and dashboards. While GreenLake offers comprehensive native monitoring capabilities, organizations can also leverage the GreenLake API to integrate with popular open-source tools like Grafana and Prometheus. This approach enables teams to consolidate monitoring data across hybrid environments, utilize existing observability workflows, and create customized dashboards tailored to specific operational needs.

Kubernetes and Helm Setup

Environment Verification

Before proceeding with the monitoring setup, verify that your Kubernetes cluster has the necessary components installed. The following shows a working environment with the GPU Operator and Prometheus monitoring stack deployed:

Services running in the gpu-operator namespace: - `gpu-operator`: Core service for GPU management (ClusterIP: 10.233.44.80:8080) - `nvidia-dcgm-exporter`: DCGM metrics exporter for Prometheus integration (ClusterIP: 10.233.15.59:9400)

Helm releases: - `gpu-operator-1753140595` (v25.3.2) in the `gpu-operator` namespace - `kube-prometheus-stack` (76.3.0) in the `monitoring` namespace

You can verify your setup using the following commands:

```
wsl=> k get svc -n gpu-operator
NAME                   TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
gpu-operator           ClusterIP   10.233.44.80   <none>        8080/TCP   78d
nvidia-dcgm-exporter   ClusterIP   10.233.15.59   <none>        9400/TCP   78d

wsl=> helm list -A
NAME                     NAMESPACE       REVISION        UPDATED
                                    STATUS          CHART                               APP
    VERSION
gpu-operator-1753140595 gpu-operator    4               2025-08-14 19:20:42.329819669 -0700
    MST deployed        gpu-operator-v25.3.2            v25.3.2
kube-prometheus-stack   monitoring      5               2025-08-15 13:06:31.169338089 -0700
    MST deployed        kube-prometheus-stack-76.3.0    v0.84.1
hjma@HSTHJMA02:~
```

External Access Configuration

The cluster has been configured with NodePort services to enable external access to Grafana and Prometheus:

```
wsl=> k get svc --field-selector spec.type=NodePort
NAME                                 TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)
                              AGE
kube-prometheus-stack-grafana        NodePort   10.233.22.241   <none>        80:30080/TCP
                               55d
kube-prometheus-stack-prometheus     NodePort   10.233.8.106    <none>        9090:30090/TCP
    ,8080:30398/TCP   55d
```

GPU utilization simulation

To simulate GPU load and verify monitoring functionality, we deployed a test pod running the gpu-burn utility. This tool performs intensive GPU computations, allowing us to observe GPU utilization metrics in our monitoring dashboards.

The following YAML manifest creates a pod that clones the gpu-burn repository, compiles it, and runs continuous GPU stress testing:

```
apiVersion: v1
kind: Pod
metadata:
    name: gpu-burn
spec:
    containers:
        - name: gpu-burn
          image: nvidia/cuda:12.2.0-devel-ubuntu22.04
          command: ["/bin/bash", "-c"]
          args:
              - |
                apt update && apt install -y git build-essential && \
                git clone https://github.com/wilicc/gpu-burn.git && \
                cd gpu-burn && make && ./gpu_burn 999999
          resources:
              limits:
                  nvidia.com/gpu: 1
    restartPolicy: Never
```

Key configuration details: - **Base image**: `nvidia/cuda:12.2.0-devel-ubuntu22.04` provides the CUDA development environment - **GPU allocation**: `nvidia.com/gpu: 1` requests a single GPU from the cluster - **Runtime**: `gpu_burn 999999` runs for approximately 277 hours (effectively continuous) - **Restart policy**: `Never` ensures the pod completes its run without automatic restarts

Deploy the pod using:

```
kubectl apply -f gpu-burn.yaml
```