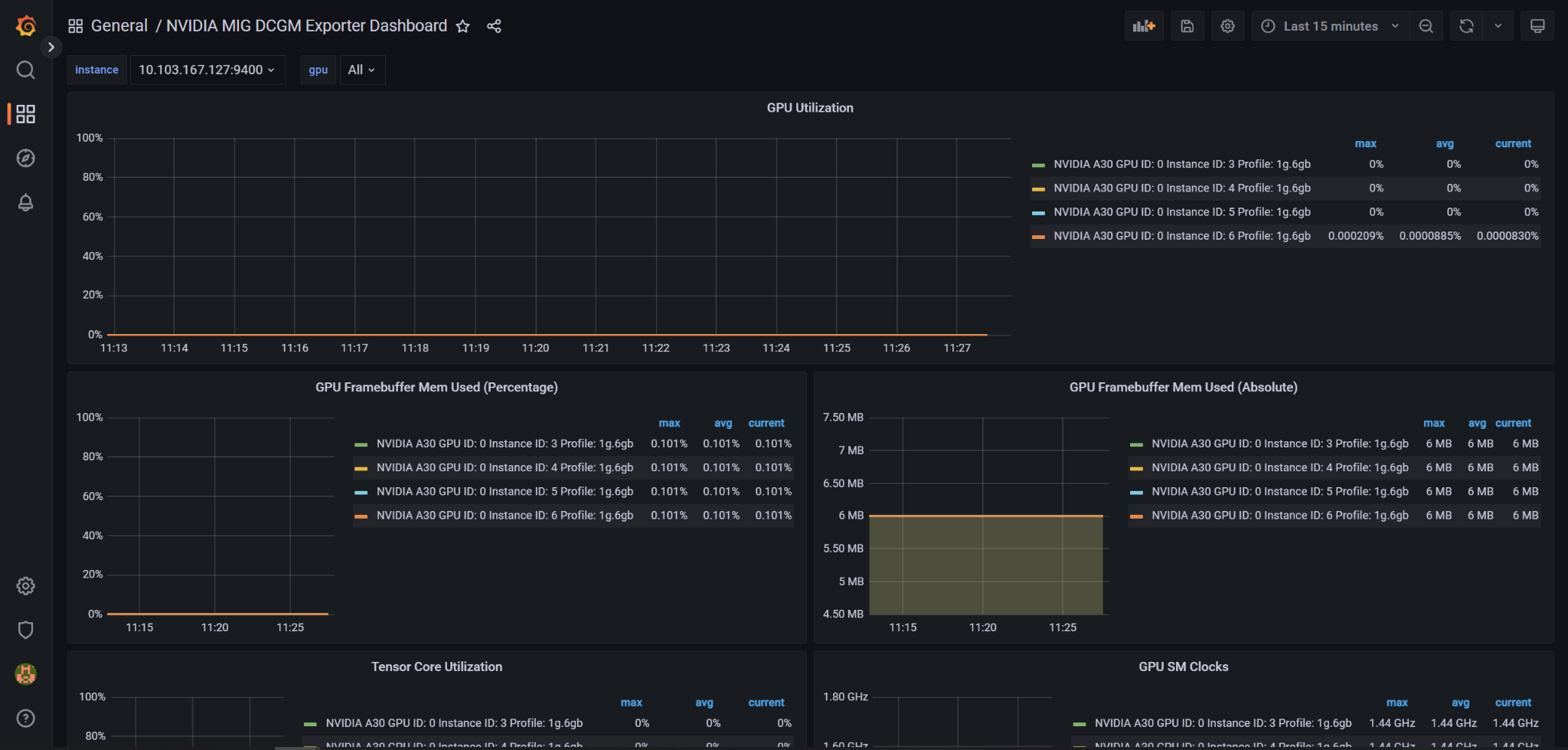
Prometheus/Grafana Deployment with DCGM Integration

# Introduction

This document outlines the concept, design, and usage of a helm chart that deploys prometheus and grafana on a Kubernetes cluster and integrates them with DCGM. The deployment also includes multiple grafana dashboards for different environments in use with DCGM (MIG, vGPU, ect.). Here is an example screenshot of the MIG dashboard:



Each MIG instance’s metrics is reported and labeled with its instance ID, here you can see some of the metrics displayed such as GPU utilization and memory usage. The github repository for this deployment can be found at: [MattFeinberg/GPU-Telemetry-DCGM (github.com)](https://github.com/MattFeinberg/GPU-Telemetry-DCGM)

## Assumptions, Constraints, Dependencies

* This deployment assumes that the NVIDIA GPU Operator has been deployed on the cluster.
  + In particular, the dcgm-exporter pod and ClusterIP service must be up and running.
* This deployment is dependent on the Kube Prometheus Stack as a subchart.
  + [helm-charts/charts/kube-prometheus-stack at main · prometheus-community/helm-charts (github.com)](https://github.com/prometheus-community/helm-charts/tree/main/charts/kube-prometheus-stack)

# Deploying Prometheus/Grafana

To deploy prometheus and grafana, this helm chart uses the Kube Prometheus Stack helm chart as a subchart. More info on this chart can be found here: [helm-charts/charts/kube-prometheus-stack at main · prometheus-community/helm-charts (github.com)](https://github.com/prometheus-community/helm-charts/tree/main/charts/kube-prometheus-stack). The subchart is added in the Chart.yaml file:

| dependencies:  - name: kube-prometheus-stack  repository: https://prometheus-community.github.io/helm-charts  version: 37.2.0 |
| --- |

Settings for the subchart are specified in the values.yaml file, which can be seen in the [design details](#_rpoyutn4x5rv).

# Integration with DCGM

To integrate the prometheus/grafana deployment with DCGM, this helm chart adds an additional prometheus scrape job to collect the metrics exposed by the dcgm-exporter:

| - job\_name: 'DCGM'  static\_configs:  - targets:  - <DCGM-clusterIP>:<dcgm-port> |
| --- |

Where <DCGM-clusterIP> and <dcgm-port> form the address at which DCGM exposes GPU metrics. This address is obtained from examining the output of kubectl get svc -A :

(example output)

| NAMESPACE NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE  default kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 10d  kube-system kube-dns ClusterIP 10.96.0.10 <none> 53/UDP,53/TCP,9153/TCP 10d  nvidia-gpu-operator gpu-operator ClusterIP 10.104.254.83 <none> 8080/TCP 10d  nvidia-gpu-operator gpu-operator-1657565362-node-feature-discovery-master ClusterIP 10.104.103.231 <none> 8080/TCP 10d  nvidia-gpu-operator nvidia-dcgm-exporter ClusterIP 10.107.190.69 <none> 9400/TCP 10d |
| --- |

By examining the nvidia-dcgm-exporter service, we can see that in this case, <DCGM-clusterIP> would be 10.107.190.69, and <dcgm-port> would be 9400. To add the additional scrape rule for prometheus, first we need the ClusterIP and port numbers, and then we can specify these values at install time with helm install. See [installation](#_axkr5k87zooh) for more details.

# Design Details

This deployment is essentially a wrapper for the Kube Prometheus Stack helm chart ([helm-charts/charts/kube-prometheus-stack at main · prometheus-community/helm-charts (github.com)](https://github.com/prometheus-community/helm-charts/tree/main/charts/kube-prometheus-stack)), which handles deploying prometheus and grafana on the cluster. Using the Kube Prometheus Stack as a subchart, this deployment then overrides and adds certain values (in values.yaml) to configure the deployment to integrate with DCGM.

**Note:** The following two subsections are just explanations of the configurations that are already included in the helm chart. These are not steps to follow.

## Prometheus Configuration

To configure prometheus, we need to expose the service using NodePort (so that we can access prometheus from a web browser). To do this, the chart’s values.yaml includes this section:

| kube-prometheus-stack:  prometheus:  service:  type: NodePort |
| --- |

The default NodePort in use is 30090, which is specified in the Kube Prometheus Stack subchart, and not overridden here. To finalize the prometheus configuration, we will need to add an additional scrape config to retrieve the metrics from DCGM. This step is done at [installation](#_axkr5k87zooh) time using helm install, and will render the values.yaml file to include the scrape rule:

| - job\_name: 'DCGM'  static\_configs:  - targets:  - <DCGM-clusterIP>:<dcgm-port> |
| --- |

## Grafana Configuration

To configure Grafana, we need to expose the service using NodePort and set the admin password. To do this, the charts values.yaml file is updated to include this section:

| grafana:  service:  type: NodePort  nodePort: 32322  defaultDashboardsEnabled: false  adminPassword: admin |
| --- |

These settings make grafana available from outside the cluster at port 32322. The grafana dashboards can be easily imported based on your cluster set up, see [Import Dashboards](#_r6spxaup952l) for more information on importing dashboards.

# Installation

Before installing, ensure that the NVIDIA GPU Operator has been installed onto your cluster. Additionally, privileges to create and deploy in a new namespace on the cluster must be set. Some distributions may require extra steps to ensure this; for example, tanzu environments require a security policy for new namespaces, like this:

| apiVersion: rbac.authorization.k8s.io/v1  kind: RoleBinding  metadata:  name: psp:vmware-system-privileged:monitoring  namespace: monitoring  roleRef:  apiGroup: rbac.authorization.k8s.io  kind: ClusterRole  name: psp:vmware-system-privileged  subjects:  - kind: Group  apiGroup: rbac.authorization.k8s.io  name: system:serviceaccounts |
| --- |

Once you are ready, go through the following steps to install the helm chart:

## Download Helm Chart

First, retrieve the helm chart by cloning the repository and cd-ing into it using:

| git clone https://github.com/MattFeinberg/GPU-Telemetry-DCGM.git  cd GPU-Telemetry-DCGM |
| --- |

## Retrieve DCGM Exporter Address

Next, we need the address where the DCGM exporter exposes GPU metrics. To obtain this address, run:

| kubectl get svc -A | sed -n -e '1p' -e '/nvidia-dcgm-exporter/p' |
| --- |

This will display the nvidia-dcgm-exporter service, which is deployed by the GPU operator. The output of the above command will print the top line (header) of kubectl get and the sed command filters the services to obtain the DCGM service. The output should look something like this:

| NAMESPACE NAME TYPE CLUSTER-IP PORT(S)  nvidia-gpu-operator nvidia-dcgm-exporter ClusterIP 10.107.190.69 9400/TCP |
| --- |

Take note of the ClusterIP and port values as they will be needed later. In the example output, the ClusterIP is 10.107.190.69 and the port is 9400. If you want, you may store them in a variable for use later as a single address:

(bash, using example values)

| address="10.107.190.69:9400" |
| --- |

This is the address where prometheus is able to access DCGM metrics, so we are now ready to install the helm chart.

## Install The Helm Chart

Ensure that you are in the GPU-Telemetry directory (created from cloning the repository) and that the helm chart is present, which should be captured in a directory named DCGMChart. You may confirm this by running:

| pwd && ls |
| --- |

Next, install the helm chart using this command. Be sure to substitute in the **ClusterIP** and **port** values obtained from the previous step.

| helm install dcgm-monitoring DCGMChart \  --create-namespace --namespace monitoring \  --set kube-prometheus-stack.prometheus.prometheusSpec.additionalScrapeConfigs[0].static\_configs[0].targets[0]=**<ClusterIP>**:**<Port>**,\  kube-prometheus-stack.prometheus.prometheusSpec.additionalScrapeConfigs[0].job\_name='DCGM-Metrics' |
| --- |

If you stored the IP and port values in a variable using the command from the previous step, the install command is as follows:

| helm install dcgm-monitoring DCGMChart \  --create-namespace --namespace monitoring \  --set kube-prometheus-stack.prometheus.prometheusSpec.additionalScrapeConfigs[0].static\_configs[0].targets[0]=$address,\  kube-prometheus-stack.prometheus.prometheusSpec.additionalScrapeConfigs[0].job\_name='DCGM-Metrics' |
| --- |

After installing, prometheus is accessible via http at <machine-IP>:30090, and grafana (credentials: admin/admin) is accessible at <machine-IP>:32322.

## Import Dashboards

Once you have logged onto grafana, you can import dashboards depending on your cluster GPUs:

* Standard Dashboard: [NVIDIA DCGM Exporter Dashboard dashboard for Grafana | Grafana Labs](https://grafana.com/grafana/dashboards/12239)
* MIG Dashboard: [NVIDIA MIG DCGM Exporter Dashboard dashboard for Grafana | Grafana Labs](https://grafana.com/grafana/dashboards/16640)

Be sure to select Prometheus as the data source when importing a dashboard.