# 曦云®系列通用计算 GPU

## mx-exporter Kubernetes 集群监控部署手册

CSOG-24047-020-F3_V01　　2024-09-06

沐曦专有和三级保密信息　　本文档受 NDA 管控

# 更新记录

| 版本 | 日期 | 更新说明 |
| --- | --- | --- |
| V01 | 2024-09-06 | 首次发布 |

# 目录

# 图目录

# 表目录

# 1. 概述

mx-exporter 是用于在集群环境中收集曦云®GPU 设备指标数据的工具。集群监控系统 Prometheus 可以通过 HTTP 从运行于每个节点的 mx-exporter 拉取设备指标数据。可视化工具 Grafana 将收集的 GPU 设备指标转化成易于理解的图表。

本文将介绍如何在 Kubernetes 集群中部署 Prometheus，Grafana，mx-exporter 来监控 GPU 设备。

# 2. 环境信息及新建 Namespace

## 2.1 环境信息

本文中各工具应用的版本参见下表，用户可根据实际情况选取版本。

**表 2-1 工具版本信息**

| 名称 | 版本 |
| --- | --- |
| Kubernetes | v1.24.0 |
| docker | 20.10.21 |
| containerd | 1.6.10 |
| Prometheus | v2.46.0 |
| Grafana | 10.0.3 |
| Helm | v3.9.3 |

## 2.2 导入 mx-exporter 镜像

**操作步骤**

1. 解压 mx-exporter 镜像包。

```
tar -zxvf mx-exporter.xxx.tgz
```

2. 根据主机的架构加载对应的镜像。对于 x86 架构主机，使用 amd64 后缀的镜像；对于 Arm 架构主机，使用 arm64 后缀的镜像。

```
cd mx-exporter; docker load -i mx-exporter-xx-amd64.xz
```

3. 将 mx-exporter 镜像推送到 Harbor 中，用户需要更改部署脚本中的镜像下载地址为设置的推送地址，例如 Harbor 为 mxcr.io。

```
docker login -u $user mxcr.io
docker tag mxcr.io/cloud/mx-exporter:xxx mxcr.io/$project/mx-exporter:xxx
docker push mxcr.io/$project/mx-exporter:xxx
```

# 2.3 在 Kubernetes 中新建 Namespace

### 操作步骤

1. 新建 namespace，回显信息如图 2-1 所示。

```
cd mx-exporter/deployment; kubectl create -f namespace.yaml
```

```
root@master:/home/user1/mx-exporter# cd mx-exporter/deployment;kubectl create -f namespace.yaml
namespace/metax-monitor created
```

**图 2-1 创建 namespace**

2. 查看新建的 namespace，回显信息如图 2-2 所示。

```
kubectl get namespace
```

```
root@master:/home/user1/mx-exporter/mx-exporter/deployment# kubectl get namespace
NAME                   STATUS   AGE
default                Active   25d
kube-flannel           Active   25d
kube-node-lease        Active   25d
kube-public            Active   25d
kube-system            Active   25d
kubernetes-dashboard   Active   25d
metax-monitor          Active   17s
```

**图 2-2 查看新建的 namespace**

# 3. mx-exporter 部署

本节介绍两种在 Kubernetes 上部署 mx-exporter 的方式，用户可根据需要选择一种。

## 3.1 Helm 方式部署

Helm 在 **mx-exporter/deployment/mx-exporter/helm** 内，可按需设置下表参数，在安装时用 `--set` 传入以生效。

**表 3-1 Helm 方式部署 exporter 参数**

| 参数 | 类型 | 描述 |
| --- | --- | --- |
| image.repository | string | 镜像地址，默认为 **mxcr.io/cloud/mx-exporter** |
| image.tag | string | 镜像版本号 |
| service.port | integer | 设置 mx-exporter pod 中 container 的端口，默认为 `8000` |
| gatherInterval | integer | 指标收集间隔，当 `gatherMode` 为 0 时启用，默认为 `10000ms` |
| gatherMode | integer | 默认为 `0`<br>`0`：根据设置的间隔时间收集数据，需与 `gatherInterval` 配合使用<br>`1`：当有请求时收集数据 |
| logMonitor | integer | 默认为 `0`<br>`0`：禁用监控 kernel log 功能<br>`1`：启用监控 kernel log 功能 |

用户根据 mx-exporter 镜像信息设置 `image.repository` 和 `image.tag`。

### 操作步骤（部署 mx-exporter）

1. 部署 mx-exporter，回显信息如图 3-1 所示。

```
cd mx-exporter/deployment/mx-exporter/helm; helm install metax-mx-exporter mx-exporter -n metax-monitor --set image.repository=xxx --set image.tag=xxxx
```

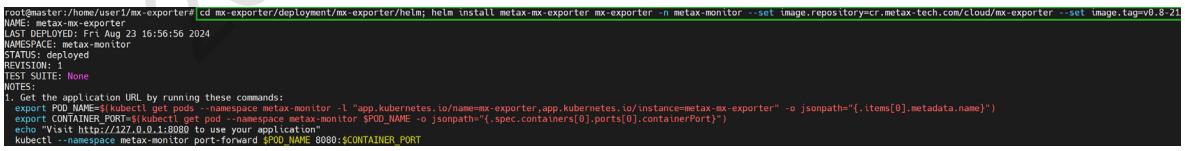


**图 3-1 Helm 方式部署 mx-exporter**

2. 查看新建资源信息，回显信息如图 3-2 所示。

```
helm list -n metax-monitor
kubectl get all -n metax-monitor -o wide
```

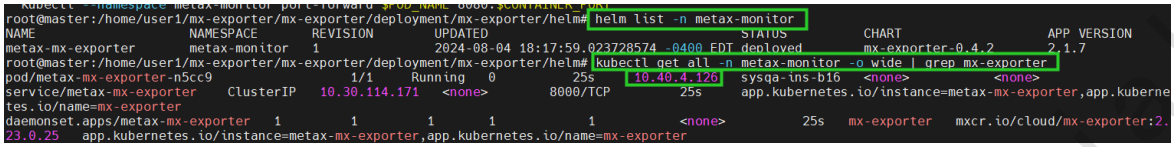


**图 3-2 查看新建资源信息**

默认每个节点都已部署 mx-exporter。需记录 mx-exporter pod IP，在 Grafana 后续展示中用于筛选需要展示指标的目标服务器。

## 查看抓取信息

mx-exporter 部署成功后，等待 40 秒，可在 k8s 中用 `curl` 命令查看 mx-exporter 抓取的 GPU 信息，回显信息如图 3-3 所示。

```
curl <mx-exporter_pod_ip>:<mx-exporter_service_port>
```

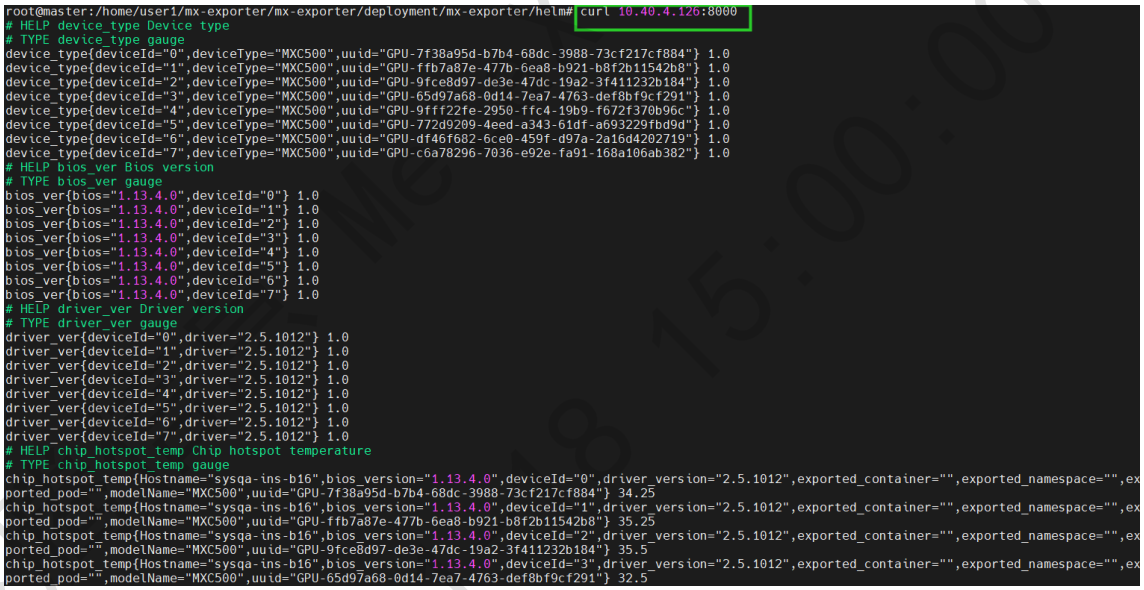


**图 3-3 查看 mx-exporter 抓取信息（Helm）**

## 删除资源

如需删除资源，可使用以下命令：

```
helm uninstall metax-mx-exporter -n metax-monitor
```

## 3.2 YAML 方式部署

如需修改下表参数，可编辑 mx-exporter/deployment/mx-exporter/mx-exporter-daemonset.yaml。

**表 3-2 YAML 方式部署 exporter 参数**

| 参数 | 类型 | 描述 |
|---|---|---|
| image | string | 镜像地址：Tag 号，用户根据导入的 mx-exporter 镜像信息设置 |
| -c | string | 用户自定义的指标配置文件，默认在 /etc/config/metrics |
| -p | integer | port，设置端口，默认为 8000 |
| -i | integer | interval，指标收集间隔，在 containers.args 中；-m 为 0 时启用，默认为 10000ms |
| -m | integer | 收集模式，默认为 0，在 containers.args 中<br>0：根据设置的间隔时间收集数据，需与-i 配合使用<br>1：当有请求时收集数据 |
| -lm | integer | kernel log 监控开关<br>0：禁用监控 kernel log 功能<br>1：启用监控 kernel log 功能 |

### 操作步骤（部署 mx-exporter）

1. 部署 mx-exporter，回显信息如图 3-4 所示。

```
cd mx-exporter/deployment/mx-exporter; kubectl create -f mx-exporter-daemonset.yaml
```

```
root@master:/home/user1/mx-exporter# cd mx-exporter/deployment/mx-exporter; kubectl create -f mx-exporter-daemonset.yaml
serviceaccount/metax-mx-exporter created
configmap/exporter-metrics-config-map created
service/metax-mx-exporter created
daemonset.apps/metax-mx-exporter created
```

**图 3-4 YAML 方式部署 mx-exporter**

2. 查看新建资源信息，回显信息如图 3-5 所示。

```
kubectl get all -n metax-monitor -o wide
```

```
root@master:/home/user1/mx-exporter/mx-exporter/deployment/mx-exporter# kubectl get all -n metax-monitor -o wide
NAME                          READY   STATUS    RESTARTS   AGE   IP             NODE           NOMINATED NODE   READINESS GATES
pod/metax-mx-exporter-cn5qg   1/1     Running   0          34s   10.40.4.114    sysqa-ins-b16   <none>           <none>

NAME                          TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)    AGE   SELECTOR
service/metax-mx-exporter     ClusterIP   10.30.4.73    <none>        8000/TCP   34s   app.kubernetes.io/instance=metax,app.kubernetes.io/name=mx-exporter

NAME                                 DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE   CONTAINERS    IMAGES                                SELECTOR
daemonset.apps/metax-mx-exporter     1         1         1       1            1           <none>          34s   mx-exporter   mxcr.io/cloud/mx-exporter:2.23.0.25   app.kubernetes.io/instance=metax,app.kubernetes.io/name=mx-exporter
```

**图 3-5 查看新建资源信息**

默认每个节点都已部署 mx-exporter。记录 mx-exporter pod IP，在 Grafana 后续展示中用于筛选需要展示指标的目标服务器。

## 查看抓取信息

mx-exporter 部署成功后，等待 40 秒，可在 k8s 中用 `curl` 命令查看 mx-exporter 抓取的 GPU 信息，回显信息如图 3-6 所示。

```
curl <mx-exporter_pod_ip>:<mx-exporter_service_port>
```

```
root@master:/home/user1/mx-exporter/mx-exporter/deployment# curl 10.40.4.114:8000
# HELP device_type Device type
# TYPE device_type gauge
device_type{deviceId="0",deviceType="MXC500",uuid="GPU-7f38a95d-b7b4-68dc-3988-73cf217cf884"} 1.0
device_type{deviceId="1",deviceType="MXC500",uuid="GPU-ffb7a87e-477b-6ea8-b921-b8f2b11542b8"} 1.0
device_type{deviceId="2",deviceType="MXC500",uuid="GPU-9fce8d97-de3e-47dc-19a2-3f411232b184"} 1.0
device_type{deviceId="3",deviceType="MXC500",uuid="GPU-65d97a68-0d14-7ea7-4763-def8bf9cf291"} 1.0
device_type{deviceId="4",deviceType="MXC500",uuid="GPU-9fff22fe-2950-ffc4-19b9-f672f370b96c"} 1.0
device_type{deviceId="5",deviceType="MXC500",uuid="GPU-772d9209-4eed-a343-61df-a693229fbd9d"} 1.0
device_type{deviceId="6",deviceType="MXC500",uuid="GPU-df46f682-6ce0-459f-d97a-2a16d4202719"} 1.0
device_type{deviceId="7",deviceType="MXC500",uuid="GPU-c6a78296-7036-e92e-fa91-168a106ab382"} 1.0
# HELP bios_ver Bios version
# TYPE bios_ver gauge
bios_ver{bios="1.13.4.0",deviceId="0"} 1.0
bios_ver{bios="1.13.4.0",deviceId="1"} 1.0
bios_ver{bios="1.13.4.0",deviceId="2"} 1.0
bios_ver{bios="1.13.4.0",deviceId="3"} 1.0
bios_ver{bios="1.13.4.0",deviceId="4"} 1.0
bios_ver{bios="1.13.4.0",deviceId="5"} 1.0
bios_ver{bios="1.13.4.0",deviceId="6"} 1.0
bios_ver{bios="1.13.4.0",deviceId="7"} 1.0
# HELP driver_ver Driver version
# TYPE driver_ver gauge
driver_ver{deviceId="0",driver="2.5.1012"} 1.0
driver_ver{deviceId="1",driver="2.5.1012"} 1.0
driver_ver{deviceId="2",driver="2.5.1012"} 1.0
driver_ver{deviceId="3",driver="2.5.1012"} 1.0
driver_ver{deviceId="4",driver="2.5.1012"} 1.0
driver_ver{deviceId="5",driver="2.5.1012"} 1.0
driver_ver{deviceId="6",driver="2.5.1012"} 1.0
driver_ver{deviceId="7",driver="2.5.1012"} 1.0
# HELP chip_hotspot_temp Chip hotspot temperature
# TYPE chip_hotspot_temp gauge
chip_hotspot_temp{Hostname="sysqa-ins-b16",bios_version="1.13.4.0",deviceId="0",driver_version="2.5.1012",exported_container="",exported_n
amespace="",exported_pod="",modelName="MXC500",uuid="GPU-7f38a95d-b7b4-68dc-3988-73cf217cf884"} 35.75
chip_hotspot_temp{Hostname="sysqa-ins-b16",bios_version="1.13.4.0",deviceId="1",driver_version="2.5.1012",exported_container="",exported_n
amespace="",exported_pod="",modelName="MXC500",uuid="GPU-ffb7a87e-477b-6ea8-b921-b8f2b11542b8"} 36.75
chip_hotspot_temp{Hostname="sysqa-ins-b16",bios_version="1.13.4.0",deviceId="2",driver_version="2.5.1012",exported_container="",exported_n
amespace="",exported_pod="",modelName="MXC500",uuid="GPU-9fce8d97-de3e-47dc-19a2-3f411232b184"} 37.25
chip_hotspot_temp{Hostname="sysqa-ins-b16",bios_version="1.13.4.0",deviceId="3",driver_version="2.5.1012",exported_container="",exported_n
amespace="",exported_pod="",modelName="MXC500",uuid="GPU-65d97a68-0d14-7ea7-4763-def8bf9cf291"} 34.5
chip_hotspot_temp{Hostname="sysqa-ins-b16",bios_version="1.13.4.0",deviceId="4",driver_version="2.5.1012",exported_container="",exported_n
```

**图 3-6 查看 mx-exporter 抓取信息（YAML）**

## 删除资源

如需删除资源，可使用以下命令：

```
cd mx-exporter/deployment/mx-exporter; kubectl delete -f .
```

# 4. Prometheus 部署

如需修改下表参数，可编辑 **mx-exporter/deployment/prometheus/prometheus-deployment.yaml**。

**表 4-1 Prometheus 部署参数**

| 参数 | 描述 |
| --- | --- |
| image | 镜像地址：Tag 号，可从 Docker Hub 下载 **prom/prometheus:v2.46.0** |

## 操作步骤（部署 Prometheus）

1. 部署 Prometheus，回显信息如图 4-1 所示。

```
cd mx-exporter/deployment/prometheus; kubectl create -f .
```

```
ot@master:/home/user1/mx-exporter# cd mx-exporter/deployment/prometheus; kubectl create -f .
usterrole.rbac.authorization.k8s.io/prometheus created
usterrolebinding.rbac.authorization.k8s.io/prometheus created
nfigmap/prometheus-server-conf created
ployment.apps/prometheus-deployment created
rvice/prometheus-service created
```

**图 4-1 部署 Prometheus**

2. 查看新建资源信息，回显信息如图 4-2 所示。

```
kubectl get all -n metax-monitor -o wide | grep prom
```

```
service/prometheus-service created
root@master:/home/user1/mx-exporter/mx-exporter/deployment/prometheus# kubectl get all -n metax-monitor -o wide | grep prom
pod/prometheus-deployment-575cc55d7f-m6s5n   1/1     Running   0          64s     10.40.4.115   sysqa-ins-b16   <none>           <none>
service/prometheus-service   NodePort    10.30.160.98   <none>        8080:30000/TCP   64s     app=prometheus-server
deployment.apps/prometheus-deployment   1/1     1            1           64s   prometheus   mxcr.io/docker/prom/prometheus:v2.46.0   app=p
rometheus-server
replicaset.apps/prometheus-deployment-575cc55d7f   1         1         1       64s   prometheus   mxcr.io/docker/prom/prometheus:v2.46.0
 app=prometheus-server,pod-template-hash=575cc55d7f
```

**图 4-2 查看新建资源信息**

可以看到 pod，service，deployment，replicaset 资源已成功创建。

## 查看抓取信息

浏览器中输入 *http://<k8s_master_ip>:<prometheus_service_port>* 访问 Prometheus GUI，其中 *prometheus_service_port* 默认为 30000。输入已知的 GPU 指标，查看抓取的信息，如图 4-3 所示。

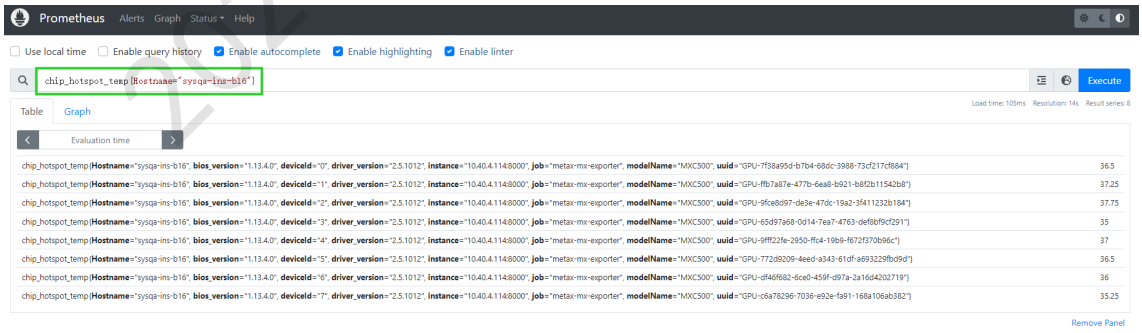


**图 4-3 在 Prometheus 中查看指标**

### 删除资源

如需删除资源，可使用以下命令：

```
cd mx-exporter/deployment/prometheus; kubectl delete -f .
```

# 5. Grafana 部署

如需修改下表参数，可编辑 **mx-exporter/deployment/grafana/deployment.yaml**。

**表 5-1 Grafana 部署参数**

| 参数 | 描述 |
| --- | --- |
| image | 镜像地址：Tag 号，可从 Docker Hub 下载 **grafana/grafana:10.0.3**。 |

## 操作步骤（部署 Grafana）

1. 部署 Grafana，回显信息如图 5-1 所示。

```
cd mx-exporter/deployment/grafana; kubectl create -f .
```

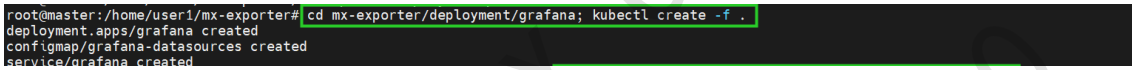


**图 5-1 部署 Grafana**

2. 查看新建资源信息，回显信息如图 5-2 所示。

```
kubectl get all -n metax-monitor -o wide | grep grafana
```

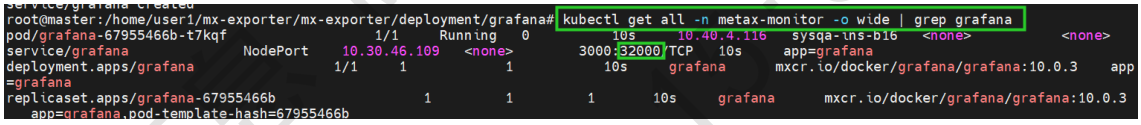


**图 5-2 查看新建资源信息**

可以看到 pod，service，deployment，replicaset 资源已成功创建。

## 删除资源

如需删除资源，可使用以下命令：

```
cd mx-exporter/deployment/grafana; kubectl delete -f .
```

## 登录 Grafana UI 界面

不同版本的 Grafana UI 显示可能有差别，图 5-3 以 10.0.3 为示例。在浏览器中输入 *http://<k8s_master_ip>:<grafana_service_port>*，其中 *grafana_service_port* 默认为 32000。首次登录用户名密码为 admin/admin。登录后可立即更新密码，也可 skip 暂时跳过。

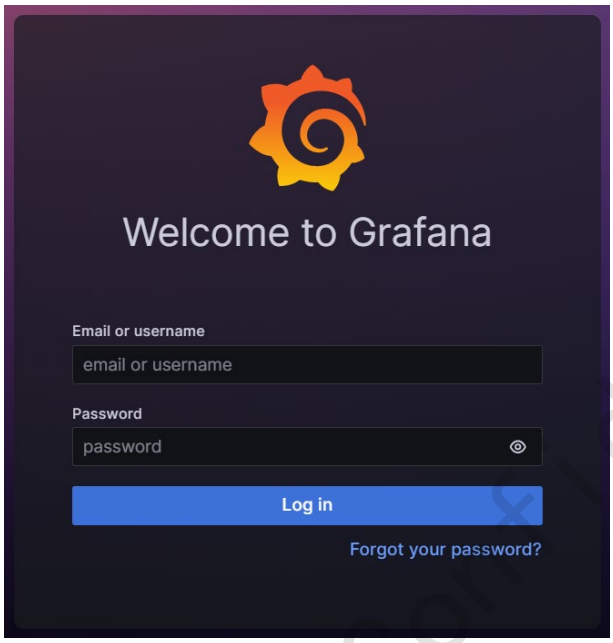


**图 5-3 Grafana 登录界面**

# 6. GPU 性能指标展示

## 6.1 接入 Prometheus 数据

### 操作步骤

1. 在 Grafana 中点击 图标，选择 **Administration > Data sources**，如图 6-1 所示。

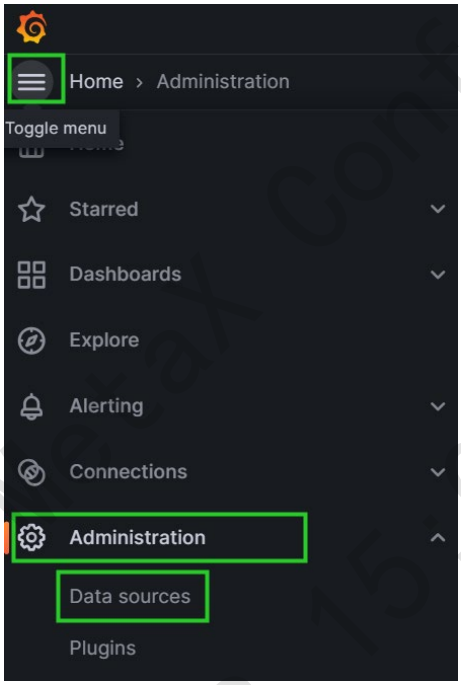


**图 6-1 Grafana 选择数据源**

2. 默认已添加 Prometheus，若未添加可点击右上角 **Add new data source**，选择添加 Prometheus。之后点击 **prometheus** 进行配置，如图 6-2 所示。

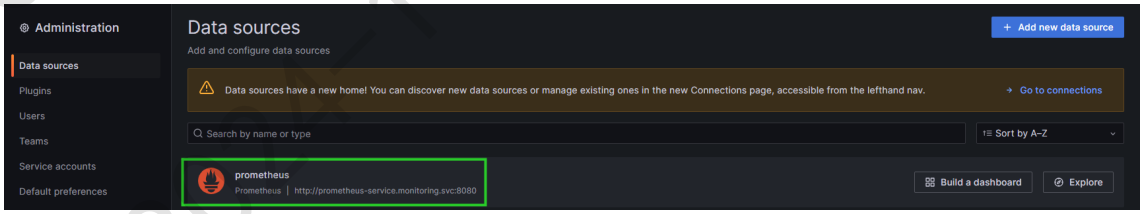


**图 6-2 Grafana 添加数据源**

3. 配置 URL 为 *http://<k8s_master_ip>:<prometheus_service_port>*，其中 *prometheus_service_port* 默认为 **30000**。点击底部 **Save & test** 测试 Prometheus 连通性，提示添加数据库成功，如图 6-3 所示。

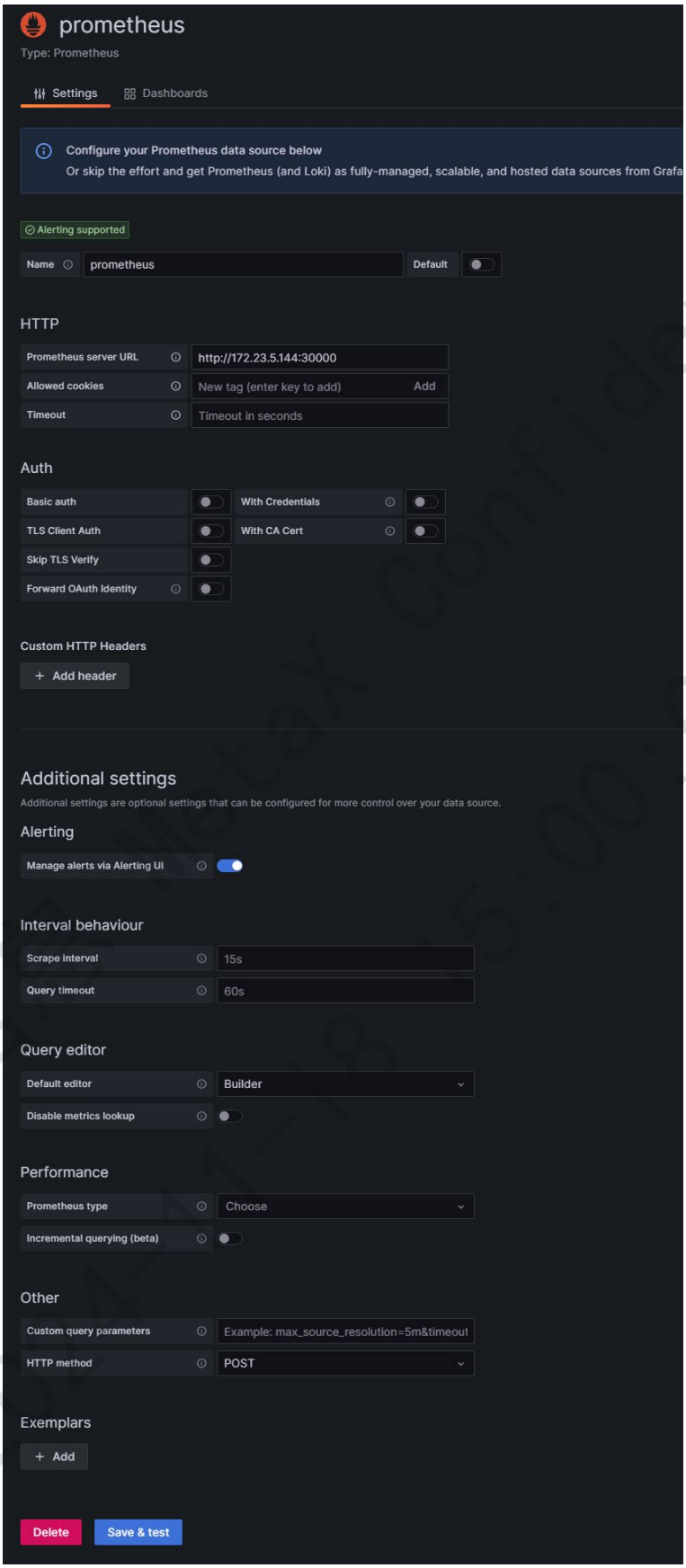


**图 6-3 Grafana 配置数据源 URL**

## 6.2 添加监控模板

### 操作步骤

1. 点击 **Dashboards** 图标，点击 **New**，选择 **Import**，如图 6-4 所示。

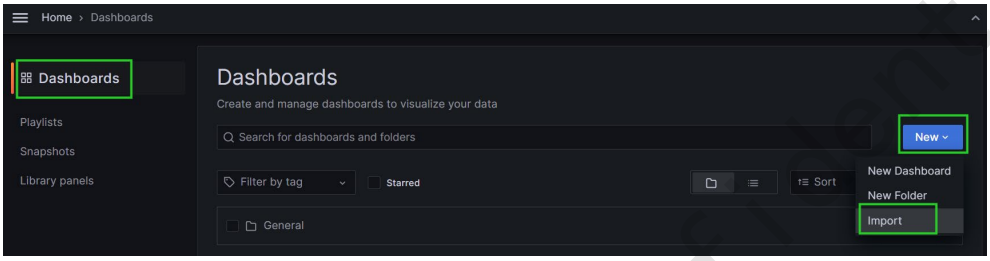


**图 6-4 Grafana 选择导入**

2. 点击 **Upload dashboard JSON file**，上传 **mx-exporter/deployment/grafana-dashboard/MetaX-GPU.json** 文件，如图 6-5 所示。

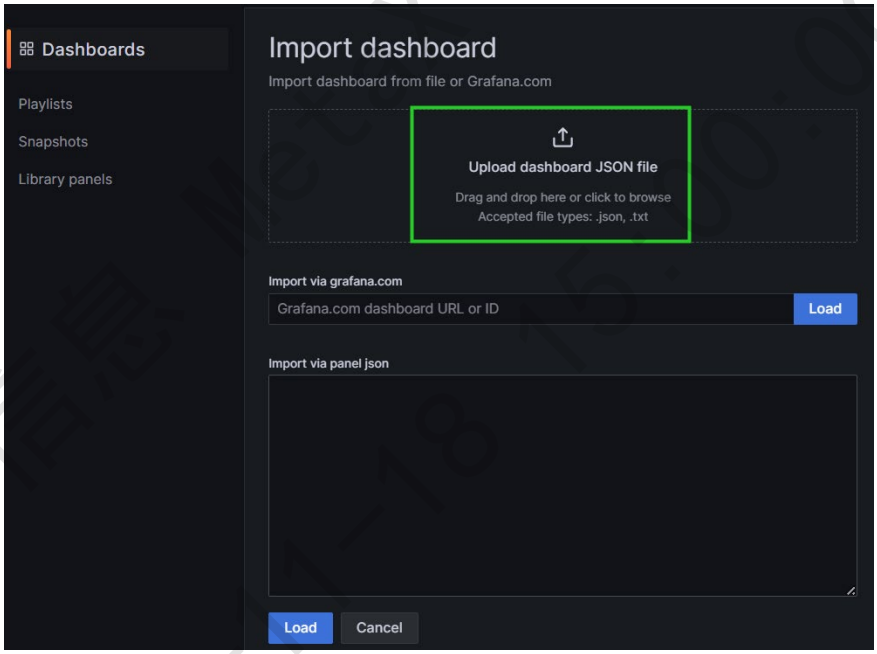


**图 6-5 Grafana 选择监控模板**

3. 数据源选 **prometheus**，点击 **Import**，如图 6-6 所示。

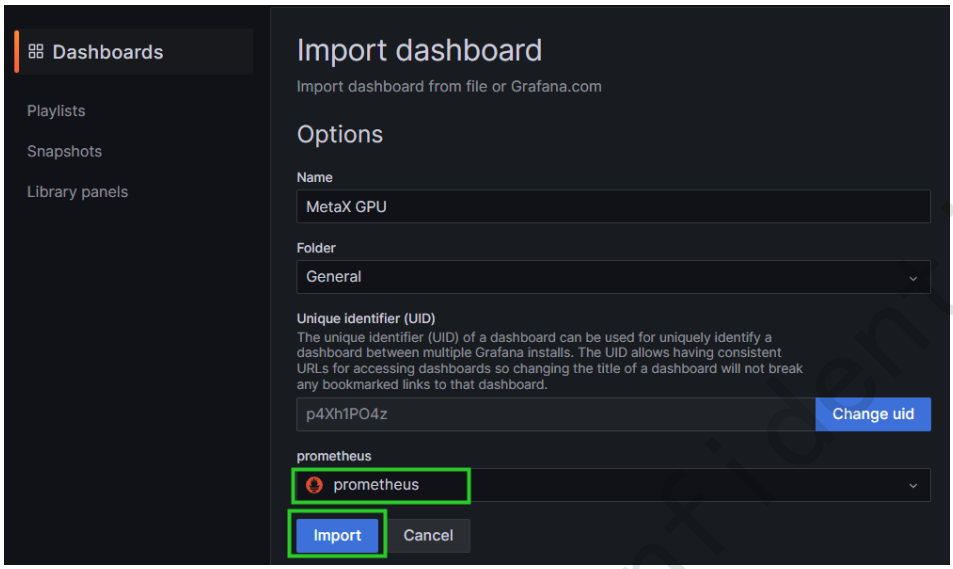


**图 6-6 Grafana 导入模板**

4. Dashboard 中 **server** 选择 *<mx-exporter_pod_ip>:端口号*，其中端口号默认为 **8000**。**device** 选择一个有效值，如 GPU 卡 0，可查看服务器卡 0 基本指标信息。点击右上角时间下拉框，可选择展示特定时间段的信息，如图 6-7 所示。

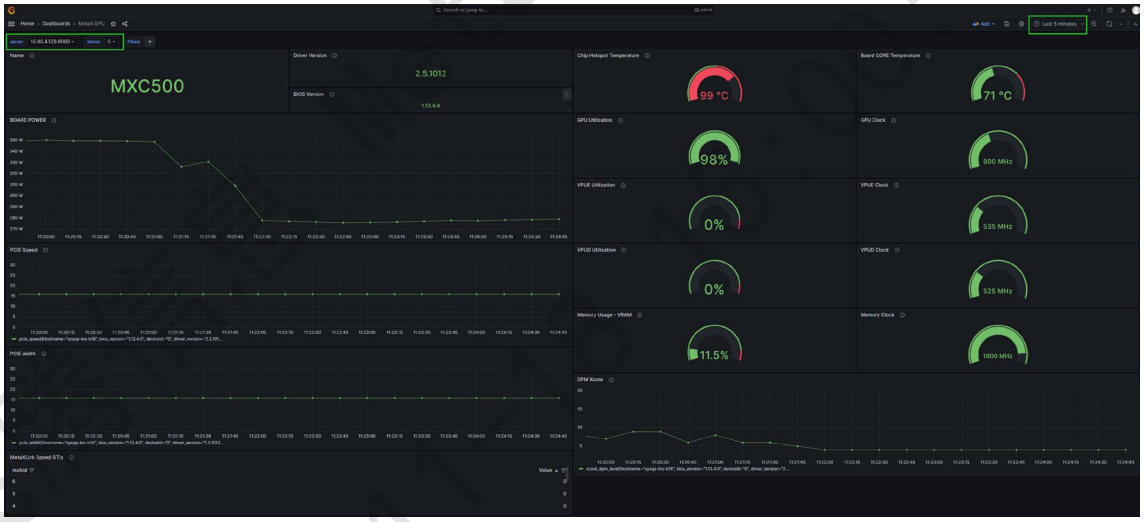


**图 6-7 Grafana 数据指标展示**

## 6.3 添加新的指标到监控面板

### 操作步骤

1. 查看 **mx-exporter/config/default-counters.csv** 中的指标信息，根据实际情况在 Grafana 中添加想要监控的指标，如 **pcie_peed**。

   有些指标仅为某些特定板卡型号中有，请注意筛选。如图 6-8 所示，第一列加“#”的内容为注释行，如果不想收集某些指标，可用“#”将其注释掉。指标标签为可用于过滤指标信息的标签，指标描述

及指标标签均可编辑（标签不可更改顺序及增删），编辑后在 Grafana 中也需做相应更改。

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| # The line begins with '#' is a comment line. | | | | | | | | | |
| # Format: metric id | metric type | metric name | metric description | label1 | label2 | | | | |
| # metric name | metric description and labels name can be modified as your need | | | | | | | | |
| # PCIe speed and width | | | | | | | | | |
| pcie_speed | Gauge | pcie_speed | Pcie current speed in GT/s | deviceId | uuid | Hostname | driver_vers | bios_versic | modelName |
| pcie_width | Gauge | pcie_width | Pcie current lanes | deviceId | uuid | Hostname | driver_vers | bios_versic | modelName |

**图 6-8 default-counters 中的指标说明**

2. 点击 **Add** 图标，下拉框中选择 **Visualization**，如图 6-9 所示。

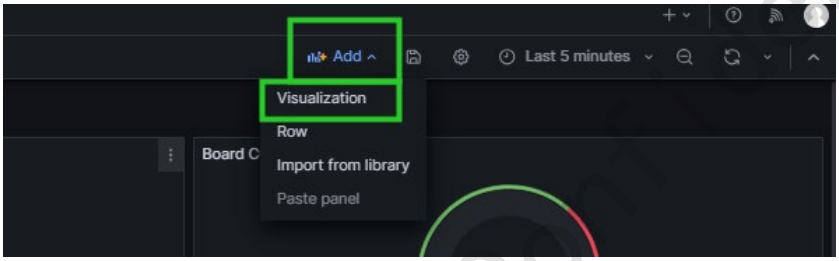


**图 6-9 Grafana 添加一个新指标图**

3. 点击数据源选择 **prometheus**，在 **Query** 中搜索指标名称及筛选的 Label，点击 **Run queries** 查看是否有数据显示，如展示符合预期，编辑右边的指标名称（**Title**）及描述（**Description**），点击 **Apply**，如图 6-10 所示。

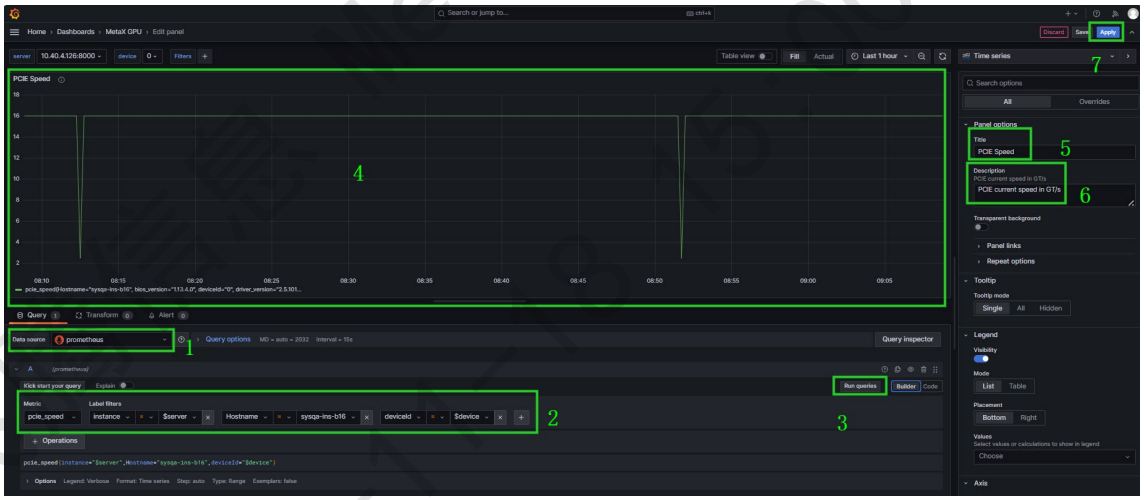


**图 6-10 Grafana 编辑新建的指标信息**

4. （可选）点击面板右侧 图标，选择 **Edit**，可再次编辑面板信息，如图 6-11 所示。

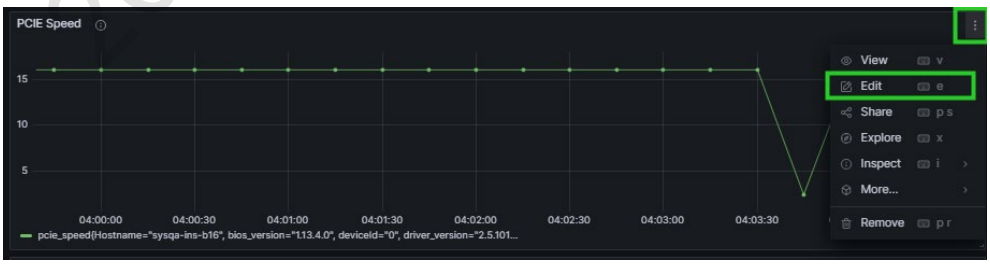


**图 6-11 Grafana 再编辑指标信息**

5. 添加完所需指标后，可点击  保存该面板，如图 6-12 所示。

   新添加的指标面板在最上面，可根据需要将光标移动到指标名，当出现可移动图标时拖拽该指标移动位置。

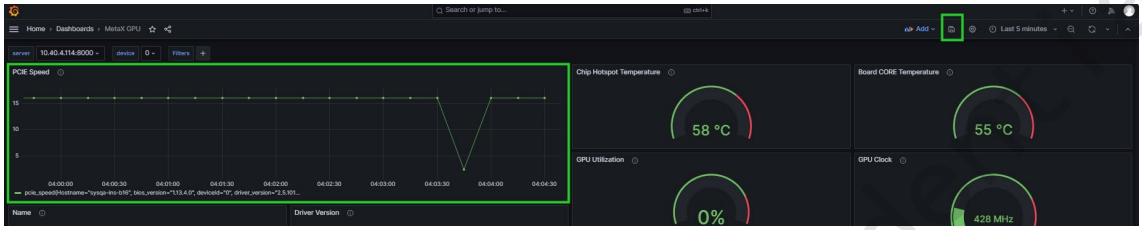


**图 6-12 Grafana 保存新建指标**

6. 在 **Details** 中描述新增或者修改的内容，点击 **Save**，如图 6-13 所示。

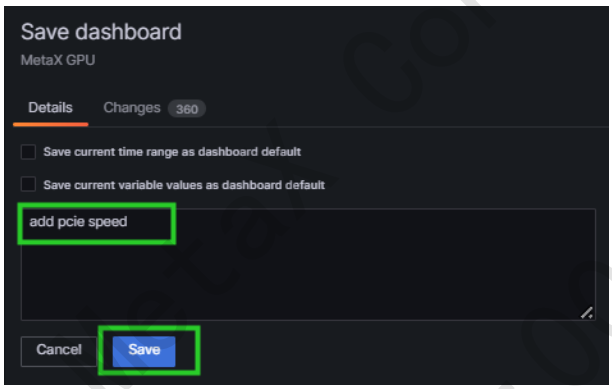


**图 6-13 Grafana 描述变更内容**

7. 点击面板左上角  ，选择 **Export > Save to file** 将当前视图保存为新的 JSON 文件，如图 6-14 所示。

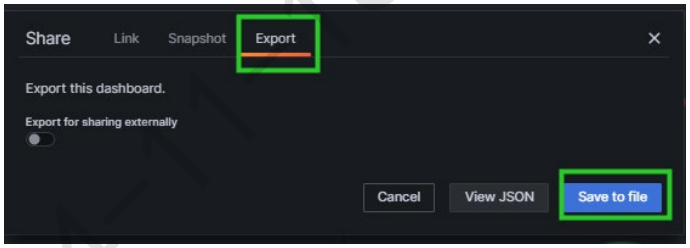


**图 6-14 Grafana 保存视图 JSON 文件**

# 7. 常见问题

## 7.1 pod 不是 Running 状态

- 可用 `kubctl describe pod <pod_name> -n metax-monitor` 查看 pod 详细信息。
- 可用 `kubectl logs <pod_name> -n metax-monitor` 查看 container log 信息。

## 7.2 导入配置文件后 Grafana 无数据显示

1. 首先需要确认 mx-exporter 有没有收集到数据，可用 `kubectl logs <mx-exporter-pod_name> -n metax-monitor` 查看 log 中有无异常。
2. 确认 **mx-exporter/mx-exporter/deployment/prometheus/config-map.yaml** 中 mx-exporter 的 job_name："metax-mx-exporter"。
3. 浏览器输入 ***<k8s_master_ip>:<prometheus_service_port>***，*prometheus_service_port* 默认为 **30000**。登录 Prometheus UI，在搜索框中输入已知指标，如 **gpu_usage**，点击 **Execute** 按钮查看 exporter 中数据是否已经导入 Prometheus 中，或者点击 **Execute** 按钮左边的 **open metrics explorer** 查询已知指标是否存在。
4. 再次确认 Grafana 指标展示界面中选择的 **server** 为 ***mx-exporter_pod_ip***，**device** 选择有效值，如 **0**。

## 声明