Node setup with Nvidia GPU

Introduction

Refer to the previous page for the general concept of Talos worker.yaml. This will build on the same concept but add required extensions and load modules needed for pods to see a GPU and use it for AI workloads. This guide will reference this documentation.

Image Factory

First, we need to create the new image that will be loaded on the node. We have the option to choose from open-source Nvidia drivers to the proprietary version. The following are the extensions you need to search for, depending on your choice. You will notice they have a lts and production version. I have been using LTS. The main item to note is that they should match. Meaning if you choose LTS chose this for both options.

Open source extensions

- `nvidia-open-gpu-kernel-modules`
- `nvidia-container-toolkit`

Proprietary extensions

- `nonfree-kmod-nvidia`
- `nvidia-container-toolkit`

Provision node

Now that we have the image file, we can add this to our worker.yaml, as seen on the previous page. This should look similar to the following under install.

```
machine:
  install:
    disk: /dev/nvme0n1 # The disk used for installations.
    image:
factory.talos.dev/installer/53b5a3efb4cca0300d7947f45577df156effe1be2f373daf3ffd8d7ba08ea899:v1.9.5
    wipe: false
```

Depending on the node type (virtual/physical), we need to boot the new node from the iso or image that came out of our image factory process. After the node is booted and ready to accept configuration, let's tell it to install these extensions and join our cluster.

```
talosctl apply-config --insecure -n <new node IP> --file worker.yaml
```

After the node reboots, we can verify the extensions are installed by running the following

```
talosctl get extensions
```

The output should look similar to the following

```
NODE           NAMESPACE   TYPE              ID          VERSION   NAME                          VERSION
192.168.249.11   runtime     ExtensionStatus   0           1         iscsi-tools                   v0.1.6
192.168.249.11   runtime     ExtensionStatus   1           1         nonfree-kmod-nvidia-lts        535.216.03-v1.9.1
192.168.249.11   runtime     ExtensionStatus   2           1         nvidia-container-toolkit-lts   535.216.03-v1.17.2
192.168.249.11   runtime     ExtensionStatus   3           1         schematic
4ba64c429e0aa252d716a668cf66b056b6ee3805f0ee0d7258a3a71e81df8e50
192.168.249.11   runtime     ExtensionStatus   modules.dep   1       modules.dep                   6.12.6-talos
```

Patch node

Now we need to patch the node to load the nvidia modules. Create a patch-gpu.yaml.

```
machine:
  kernel:
    modules:
      - name: nvidia
      - name: nvidia_uvm
      - name: nvidia_drm
      - name: nvidia_modeset
  sysctls:
    net.core.bpf_jit_harden: 1
```

Apply the patch with the following.

```
talosctl patch mc --patch @patch-gpu.yaml
```

Confirm the patch with the following commands.

```
talosctl read /proc/driver/nvidia/version


NVRM version: NVIDIA UNIX x86_64 Kernel Module  535.216.03  Fri Oct 25 22:43:06 UTC 2024
GCC version:  gcc version 14.2.0 (GCC)
```

```
talosctl read /proc/modules


nvidia_uvm 1908736 0 - Live 0xffffffffc4175000 (PO)
nvidia_drm 94208 0 - Live 0xffffffffc0575000 (PO)
nvidia_modeset 1531904 2 nvidia_drm, Live 0xffffffffc4356000 (PO)
nvidia 62771200 19 nvidia_uvm,nvidia_modeset, Live 0xffffffffc0596000 (PO)
```

Patch runtime

We need to set Nvidia as the default runtime. This can be done with a YAML inside of k8s, but I found the best way is to apply this to the node. I don't know why this is required, but pods would not see the GPU or even run nvidia-smi without this

create a file runtime-patch.yaml

```
- op: add
  path: /machine/files
  value:
    - content: |
        [plugins]
          [plugins."io.containerd.cri.v1.runtime"]
            [plugins."io.containerd.cri.v1.runtime".containerd]
              default_runtime_name = "nvidia"
      path: /etc/cri/conf.d/20-customization.part
      op: create
```

```
talosctl patch mc --patch @runtime-patch.yaml
```

Nvidia device plugin

We can use Helm to install the device plugin. This runs a daemonset looking for nodes with a GPU. It will only run on those nodes.

> Time slicing is not enabled by default. This means one GPU can only be used by one pod.The following configuration will enable one GPU to share time with many pods at once.

Create the config file time-slicing-config.yaml

```
version: v1
flags:
  migStrategy: none
sharing:
  timeSlicing:
    resources:
    - name: nvidia.com/gpu
      replicas: 5
```

Now, we can pass the config file to helm and install the device plugin.

```
helm repo add nvdp https://nvidia.github.io/k8s-device-plugin
helm repo update
helm install nvidia-device-plugin nvdp/nvidia-device-plugin --version=0.13.0 --set=runtimeClassName=nvidia \
--set-file config.map.config=./time-slicing-config.yaml \
--namespace nvidia-system
```

Testing

Let us run this pod to test if the GPU is picked up correctly.

```
apiVersion: v1
kind: Pod
metadata:
  name: gpu-pod
spec:
  restartPolicy: Never
  containers:
    - name: cuda-container
      image: nvcr.io/nvidia/k8s/cuda-sample:devicequery-cuda11.7.1-ubuntu20.04
      resources:
        limits:
          nvidia.com/gpu: 1 # requesting 1 GPU
  tolerations:
```

```
- key: nvidia.com/gpu
  operator: Exists
  effect: NoSchedule
```

Revision #4
Created 25 March 2025 22:20:03 by Kory
Updated 29 March 2025 15:58:21 by Kory