OpenShift on NVIDIA GPU Accelerated Clusters

Installation Guide
Table of Contents

Chapter 1. Introduction........................................................................................................ 1
Chapter 2. Requirements..................................................................................................... 2
Chapter 3. Installation..........................................................................................................5
Chapter 4. GPU Support.......................................................................................................7
  4.1. Installing via OpenShift OperatorHub................................................................. 7
  4.2. Installing via Helm................................................................................................. 12
  4.3. Installing GPU Operator via Helm................................................................. 12
  4.4. Running nvidia-smi......................................................................................... 15
  4.5. Cleanup............................................................................................................. 16
Chapter 5. Troubleshooting............................................................................................. 17
Chapter 6. References...................................................................................................... 19
Chapter 1. Introduction

Kubernetes is an open-source platform for automating the deployment, scaling, and managing of containerized applications.

Red Hat OpenShift is a security-centric and enterprise-grade hardened Kubernetes platform for deploying and managing Kubernetes clusters at scale, developed and supported by Red Hat.

Kubernetes via Red Hat OpenShift 4 includes enhancements to Kubernetes so users can easily configure and use GPU resources for accelerating workloads such as deep learning.
Chapter 2. Requirements

- **Supported Red Hat OpenShift Versions**
  
  OpenShift 4.4.29+, 4.5, and 4.6

- **Red Hat Subscription**
  
  Installing and running OpenShift requires a Red Hat account and additional subscriptions.

- **Internet Access**
  
  To perform subscription management, including legally entitling your software from Red Hat, your systems must have direct internet access to install the cluster.

- **Hardware**

<table>
<thead>
<tr>
<th>System Type</th>
<th>Description</th>
<th>Minimum Number of Systems</th>
<th>Recommended Specs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deployment system</td>
<td>System for executing the deployment.</td>
<td>BYO</td>
<td>Any Mac OS or Linux system with 300MB of disk space</td>
</tr>
<tr>
<td>Load balancer</td>
<td>The load balancer services allow external access to the OpenShift Container Platform cluster and distributes the work across various nodes of the cluster.</td>
<td>BYO or 1</td>
<td>Xeon Gold 5118 [12 cores, 16.5MB cache, 2.3Ghz/3.2Ghz] / 32GB RAM [distributed] / Single 40/50GbE NIC / 800GB SAS SSD</td>
</tr>
<tr>
<td>Bootstrap</td>
<td>Because each machine in the cluster requires information about the cluster when it is provisioned, the OpenShift Container Platform uses a temporary bootstrap machine during initial configuration to</td>
<td>1 (Temporary)</td>
<td></td>
</tr>
<tr>
<td>System Type</td>
<td>Description</td>
<td>Minimum Number of Systems</td>
<td>Recommended Specs</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Master (Control plane)</td>
<td>The master nodes run services that are required to control the Kubernetes cluster. In OpenShift Container Platform, the master machines are the control plane.</td>
<td>3</td>
<td>Dual Xeon Gold 5118 (12 cores, 16.5MB cache, 2.3Ghz/3.2Ghz) / 32GB RAM [distributed] / Single 40/50GbE NIC / 800GB SAS SSD</td>
</tr>
<tr>
<td>Worker (Compute)</td>
<td>The worker nodes are where the actual workloads requested by Kubernetes users run and are managed. The worker nodes advertise their capacity and the scheduler, which is part of the master services, determines on which nodes to start containers and Pods.</td>
<td>2+</td>
<td>OEM NGC-Ready T4 or V100 Systems with dual 50 GbE NICs</td>
</tr>
</tbody>
</table>

**Network Connectivity**

All systems require access to DHCP and DNS servers.

**HTTP server**

An HTTP server that is accessible from the deployment system is necessary for deploying the systems.

**PXE or iPXE server**

A PXE or iPXE server is needed for deploying the systems. Note, Red Hat also provides instructions for deployment without PXE but this method requires additional steps to set up the systems.
- **RHCOS**
  The compressed metal BIOS, UEFI, kernel, and initramfs files from the Red Hat portal are needed for deploying on the systems. These should be located on your HTTP server.

- **[Optional] NFS Storage**
  For production clusters, access to NFS is necessary for setting up a persistent volume (PV). 100 GiB minimum.
Chapter 3. Installation

The following image illustrates the process of creating the compute machines.

The following is a high-level overview of the installation process.

1. Configure networking (DHCP and system to system communication).
2. Provision two layer-4 load balancers and DNS.
3. Download the installer to the deployment system.
4. Create the installation configuration file.
5. Generate Ignition files for the systems (bootstrap, masters and workers) and upload to HTTP server.

   **Note:** The bootstrap ignition file is only good for 24 hrs due to the certificates within.

6. Configure systems and images for PXE boot.
7. Create the bootstrap, master and compute systems.
8. Create the cluster by starting the installation process cluster.
9. Remove the bootstrap machine from the load balancer.
10. Log into the cluster, export Kubernetes credentials for safe keeping, approve pending CSR
CSR certificates if necessary and confirm all operators come online.
11. Configure image registry storage.
   - For non-production servers, an empty directory can be used.
For production servers, a persistent volume (PV) is necessary (100GiB minimum, NFS).

12. The installation is complete when all components report as available and the Kubernetes API is communicating with PODS.

The detailed installation instructions are maintained by Red Hat on their documentation website.
Chapter 4. GPU Support

Note: This section assumes that an OCP cluster is up and running with a GPU worker node.

The NVIDIA GPU Operator uses the operator framework within Kubernetes to automate the management of all NVIDIA software components needed to provision the GPU. These components include the NVIDIA drivers (to enable CUDA), Kubernetes device plugin for GPUs, the NVIDIA Container Runtime, automatic node labelling, DCGM based monitoring, GPU Feature Discovery, and others.

The recommended way to set up GPU support within OpenShift is to use the GPU Operator. There are two workflows for setting up GPU support in OpenShift - via the OpenShift OperatorHub or via Helm.

4.1. Installing via OpenShift OperatorHub

The NVIDIA GPU Operator is part of the Red Hat Certified Operator Catalog and can be installed via the OperatorHub portal available in the OpenShift 4 web console.

It is recommended to have administrator privileges when installing the GPU Operator in the cluster.

1. Open the Red Hat Openshift Container Platform and, from the Operator Catalog, install NodeFeatureDiscovery(NFD) operator.
2. Once NFD is installed, create an instance of NFD through NodeFeatureDiscovery tab as below.
   a). Click Operators > Installed Operators from the side menu.

   b). Click the Node Feature Discovery entry.
c). Click Create NodeFeatureDiscovery.

This will start the NFD operator that will label the nodes in the cluster that have GPUs.

3. Before installing gpu-operator, create the gpu-operator-resources namespace.
   As shown in the screenshot, select Home > Projects from the side menu, then click Create Project and create a new project with the name "gpu-operator-resources".
   This is the namespace where the GPU Operator components will be installed.

4. Install the NVIDIA GPU Operator.
a). From the side menu, select Operators > OperatorHub, then search for the NVIDIA GPU Operator.
b). Once found, click Install.

5. Identify the nodes in the cluster that have GPUs.

From the side menu, select Operators > Installed Operators, then click Node Feature Discovery.

6. Create the cluster policy for NVIDIA GPU Operator.

a). From the side menu, select Operators > Installed Operators, then click NVIDIA GPU Operator.
b). Select the ClusterPolicy tab, then click **Create ClusterPolicy**. The platform assigns the default name “gpu-cluster-policy”.

To see that all resources deployed by **gpu-operator** in the namespace **gpu-operator-resources** are running, click **Workloads > Pods** from the side menu.
At this point, the GPU Operator will proceed and install all the required components to set up NVIDIA GPUs in the OpenShift 4 cluster.

**Note:** The nvidia-driver-validation pod may show "CreateContainerError" for a period of time while the operator is starting up the components. The status under the "Installed Operators" should change to "Succeeded" to indicate a successful installation of the GPU Operator.

### 4.2. Installing via Helm

To install Helm v3 on OpenShift, please follow the official documentation from Red Hat.

### 4.3. Installing GPU Operator via Helm

This section describes how to install GPU Operator via Helm v3.

1. Add a cluster-wide entitlement via a Kubernetes secret.
   This is required to download packages used to build the driver container.
   These instructions assume you downloaded an entitlement encoded in base64 from access.redhat.com or extracted it from an existing node.
   In the following commands, the entitlement certificate is copied to `nvidia.pem`, but it can be copied to any accessible location.

   ```bash
   $ cp <path/to/pem/file>/<certificate-file-name>.pem nvidia.pem
   ```
2. Validate the cluster-wide entitlement with a test pod that queries a Red Hat subscription repo for the kernel-devel package.

```bash
$ cat << EOF >> mypod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: cluster-entitled-build-pod
spec:
  containers:
  - name: cluster-entitled-build
    image: registry.access.redhat.com/ubi8:latest
    command: ['/bin/sh', '-c', 'dnf search kernel-devel --showduplicates']
    restartPolicy: Never
EOF

$ oc get pods -n default
$ oc logs cluster-entitled-build-pod -n default
```

3. Add and update the NVIDIA Helm repository.

```bash
$ helm repo add nvidia https://nvidia.github.io/gpu-operator
```
4. Create the default project in OpenShift.

   $ helm repo update
   $ oc project

5. Install the GPU operator.

   $ helm install gpu-operator nvidia/gpu-operator --version=1.6.2 --set platform.openshift=true,operator.validator.version=vectoradd-cuda10.2-ub8,operator.defaultRuntime=crio,nfd.enabled=false,devicePlugin.version=v0.8.2-ub8,dcmgExporter.version=2.1.4-2.2.0-ub8,toolkit.version=1.4.7-ub8 --wait

6. View the NFD pods.

   $ oc get all | egrep 'node|gpu'

   //Output
   pod/gpu-operator-1588201241-node-feature-discovery-master-748cd2vk4 1/1 Running 0 79s
   pod/gpu-operator-1588201241-node-feature-discovery-worker-2cwlw 1/1 Running 0 79s
   pod/gpu-operator-1588201241-node-feature-discovery-worker-7brcf 1/1 Running 0 79s
   pod/gpu-operator-1588201241-node-feature-discovery-worker-8grj5 1/1 Running 0 79s
   pod/gpu-operator-1588201241-node-feature-discovery-worker-8mswp 1/1 Running 0 65s
   pod/gpu-operator-1588201241-node-feature-discovery-worker-rr5lm 1/1 Running 0 79s
   pod/gpu-operator-1588201241-node-feature-discovery-worker-z64sd 1/1 Running 0 79s
   pod/gpu-operator-6bf6f99bc6-h647m 1/1 Running 0 79s
   service/gpu-operator-1588201241-node-feature-discovery-master ClusterIP 172.30.39.213 <none> 8080/TCP 79s
daemonset.apps/gpu-operator-1588201241-node-feature-discovery-worker 5 5 5 5 <none> 80s
deployment.apps/gpu-operator 1 1 1 80s
deployment.apps/gpu-operator-1588201241-node-feature-discovery-master 1 1 1 80s
replicaset.apps/gpu-operator-1588201241-node-feature-discovery-master-748c44c4df 1 1 1 80s
replicaset.apps/gpu-operator-6bf6f99bc6 1 1 1 80s

7. View the GPU device is discovered on the GPU node.

   $ oc describe node ip-10-0-137-45.us-west-1.compute.internal | egrep 'Roles|pci'

   //Output
   Roles:   worker
   feature.node.kubernetes.io/pci-10de.present=true
   feature.node.kubernetes.io/pci-1d0f.present=true

8. View the GPU operator namespace resources.

   $ oc get all -n gpu-operator-resources

   //Output
   NAME                                      READY STATUS    AGE
   pod/nvidia-container-toolkit-daemonset-sgr7h 160m 1/1 Running 0
   pod/nvidia-dcmg-exporter-twjx4 153m 2/2 Running 0
   pod/nvidia-device-plugin-daemonset-6tbfv 156m 1/1 Running 0
   pod/nvidia-device-plugin-validation 156m 0/1 Completed 0
9. Verify that the GPU Operator installation completed successfully.

The GPU Operator validates the stack through the `nvidia-device-plugin-validation` pod and `nvidia-driver-validation` pod. If both completed successfully, the stack works as expected.

```
$ oc logs nvidia-driver-validation -n gpu-operator-resources | tail

//Output
make[1]: Leaving directory '/usr/local/cuda-10.2/cuda-samples/Samples/warpAggregatedAtomicsCG'
make: Target 'all' not remade because of errors.
> Using CUDA Device [0]: Tesla T4
> GPU Device has SM 7.5 compute capability
[Vector addition of 50000 elements]
Copy input data from the host memory to the CUDA device
CUDA kernel launch with 196 blocks of 256 threads
Copy output data from the CUDA device to the host memory
Test PASSED
Done
```

```
$ oc logs nvidia-device-plugin-validation -n gpu-operator-resources | tail

//Output
make[1]: Leaving directory '/usr/local/cuda-10.2/cuda-samples/Samples/warpAggregatedAtomicsCG'
make: Target 'all' not remade because of errors.
> Using CUDA Device [0]: Tesla T4
> GPU Device has SM 7.5 compute capability
[Vector addition of 50000 elements]
Copy input data from the host memory to the CUDA device
CUDA kernel launch with 196 blocks of 256 threads
Copy output data from the CUDA device to the host memory
Test PASSED
Done
```

4.4. Running `nvidia-smi`
To view GPU utilization, run `nvidia-smi` from a pod in the GPU operator daemonset.

```bash
$ oc project gpu-operator-resources
$ oc get pods
$ oc exec -it nvidia-driver-daemonset-m7mwk nvidia-smi
```

### 4.5. Cleanup

This section describes how to clean up (remove) the GPU Operator in case it is no longer needed.

1. Get the GPU Operator instance id.
   ```bash
   $ helm ls -n default | awk '/gpu-operator/{print $1}'
   //Output
gpu-operator-1583173374
   ```
2. Delete the GPU operator instance obtained from the previous step.
   Example
   ```bash
   $ helm delete gpu-operator-1583173374 -n default
   //Output
   release "gpu-operator-1583173374" uninstalled
   ```
3. Delete the cluster policies.
   ```bash
   $ oc delete crd clusterpolicies.nvidia.com
   //output
customresourcedefinition.apiextensions.k8s.io "clusterpolicies.nvidia.com" deleted
   ```
4. [Perform this step only if using OpenShift 4.3] Delete the OpenShift 4.3 special resources.
   ```bash
   $ oc delete crd specialresources.sro.openshift.io
   //Output
customresourcedefinition.apiextensions.k8s.io "specialresources.sro.openshift.io" deleted
   ```
Chapter 5. Troubleshooting

This section includes a collection of potential errors that users may encounter when using the NVIDIA GPU Operator on OpenShift 4.

**Driver pod fails on OpenShift with dnf error**

The device driver pod has failed to start correctly with the following error in the logs:

```bash
+ dnf install -q -y elfutils-libelf.x86_64 elfutils-libelf-devel.x86_64
Error: Unable to find a match: elfutils-libelf-devel.x86_64
```

This error indicates that the UBI-based driver pod does not have subscription entitlements correctly mounted so that additional required UBI packages are not found. Please refer to this blog post on the details for entitlementing your cluster for the driver pod on non-RHEL hosts. You will need a valid subscription that is attached to a Red Hat system, even a Virtual subscription. Please refer to your account details at https://access.redhat.com/management/.

**nvidia-device-plugin-validation pod remains in Pending state**

Once the custom resource has been created, all pods appear to be healthy and Running except for the nvidia-device-plugin-validation pod which remains Pending due to no available GPU. This is due to a race condition issue with the nvidia-device-plugin pod. This can be confirmed by checking the logs of the nvidia-device-plugin:

```
2020/05/15 14:44:23 Loading NVML
2020/05/15 14:44:23 Failed to initialize NVML: could not load NVML library.
2020/05/15 14:44:23 If this is a GPU node, did you set the docker default runtime to 'nvidia'? 
2020/05/15 14:44:23 You can check the prerequisites at: https://github.com/NVIDIA/k8s-device-plugin#prerequisites
2020/05/15 14:44:23 You can learn how to set the runtime at: https://github.com/NVIDIA/k8s-device-plugin#quick-start
```

The current solution is to simply delete the nvidia-device-plugin pod and allow it to be recreated. The nvidia-device-plugin-validation pod should then eventually transition to Completed.
nvidia-container-toolkit pod in CrashLoopBackoff on OpenShift

When deploying the operator using Helm on OpenShift, the nvidia-container-toolkit pod has CrashLoopBackoff status and other pods are unhealthy also. This condition may occur if somehow the wrong container runtime has been specified for the operator. OpenShift requires that the runtime be specified as “crio” instead of “docker”. You can confirm this by checking the logs of the nvidia-container-toolkit pod for a line that reads:

```
+ RUNTIME=crio
```

If it instead indicates that the runtime is “docker”, you will need to confirm that the value of “operator.defaultRuntime=crio” is being correctly passed to the Helm chart, then uninstall and redeploy the operator.

“cross-namespace owner references are disallowed” from Helm

If the gpu-operator pod is running and a ClusterPolicy custom resource has been created but there are no pods being created, check the logs of the gpu-operator pod. If you see an error that reads “cross-namespace owner references are disallowed” you should uninstall the current operator and make sure there are no older “clustercpolicies.nvidia.com” CRD in place before Helm installs the CRD in its package. Once deleted, try the Helm install again.

Helm install fails when there is existing cluster-wide resources

Currently due to technical details, the OLM-based install of the operator on OpenShift requires that the “gpu-operator-resources” project be created in advance of installing the operator into that project. However the OLM-based install will install the CRD. If at any time there is a switch over to Helm-based installs those and other existing cluster-wide resources must be deleted first. Otherwise, the Helm install will fail with errors about the existing resource conflicts:

```
Error: rendered manifests contain a resource that already exists.
```
Chapter 6. References

- NVIDIA Cloud Native Documentation
- OpenShift 4.1 Bare Metal Installation Documentation
- OpenShift 4.2 Bare Metal Installation Documentation
- OpenShift 4.3 Bare Metal Installation Documentation
- OpenShift 4.4 Bare Metal Installation Documentation
- OpenShift 4.5 Bare Metal Installation Documentation
- OpenShift 4.6 Bare Metal Installation Documentation
- OpenShift 4.1 Architectural Overview
- OpenShift 4.2 Architectural Overview
- OpenShift 4.3 Architectural Overview
- OpenShift 4.4 Architectural Overview
- OpenShift 4.5 Architectural Overview
- OpenShift 4.6 Architectural Overview
- NVIDIA GPU Operator [NGC]
- NVIDIA GPU Operator [GitHub]
- NVIDIA blog on OpenShift Installation [for reference only; the steps are for an older version of the GPU Operator]