Enabling NVIDIA GPUs for Dell EMC Ready Stack for Red Hat OpenShift Container Platform 4.6

March 2021

H18676

Deployment Guide

Abstract

This deployment guide describes how to enable NVIDIA GPUs for use in applications running on top of Dell EMC Ready Stack for OpenShift Container Platform 4.6.

Dell Technologies Solutions

Copyright

Published in the USA 03/21 Deployment Guide H18676.

Contents

Introduction

This guide describes the integration steps that are required to enable graphics processing units (GPUs) for applications in a Red Hat OpenShift Container Platform 4.6 solution based on Dell EMC Ready Stack designs.

OpenShift Container Platform is an on-premises Kubernetes implementation that enables customers to build and manage containerized applications. OpenShift is a Red Hat distribution of the upstream OKD open source project. For more information, see OpenShift Container Platform Documentation.

Document scope

This guide describes how to:

- Enable entitled driver container builds on OpenShift
- Enable node feature discovery (NFD)
- Install and enable the NVIDIA GPU operator
- Provide GPU resources to pods

We value your feedback

Contact the Dell EMC Solutions team by email with your comments. Alternatively, contact the Dell EMC OpenShift team at openshift@dell.com.

Author: Piyush Tandon

Note: For additional information about this solution, see the Dell Technologies Solutions Info Hub for Containers.

Integration tasks

This section describes the tasks that you must perform to integrate NVIDIA GPUs on OpenShift Container Platform 4.6 built on top of a Dell EMC Ready Stack architecture.

Prerequisites

Before following the steps in this guide, ensure that your environment has:

- A working OpenShift 4.6 cluster on bare metal. If you do not have an OpenShift cluster that is operational, see the *Dell EMC Ready Stack for Red Hat OpenShift Container Platform 4.6 Deployment Guide*.
- Compute nodes that are running Red Hat CoreOS (RHCOS)
- At least one compute node with supported NVIDIA GPUs

Note: Unless otherwise specified, run the commands in this guide on the CSAH node that is managing the OpenShift cluster or on the management node for the OpenShift cluster.

The examples in this guide use `gpu-operator-resources` for the namespace.

Enabling entitled builds

The NVIDIA GPU Operator deploys several pods that are used to manage and enable GPUs for use in OpenShift Container Platform. Some of these pods require packages that are not available by default in the Universal Base Image (UBI) that OpenShift Container Platform uses. To make packages available to the NVIDIA GPU driver container, you must enable cluster-wide entitled container builds in OpenShift.

At a high level, enabling entitled builds involves three steps:

1. Download Red Hat OpenShift Container Platform subscription certificates from the Red Hat Customer Portal (access requires login credentials).
2. Create a MachineConfig that enables the subscription manager and provides a valid subscription certificate. Wait for the nodes to reboot and then finish applying the MachineConfig.
3. Validate that entitled builds are enabled.

The following sections elaborate on these steps. For more information about entitled builds in OpenShift, see this Red Hat blog post.

Download the subscription certificates

Obtain your subscription certificates under the **Subscriptions** tab in Red Hat Customer Portal, as shown in the following figure:

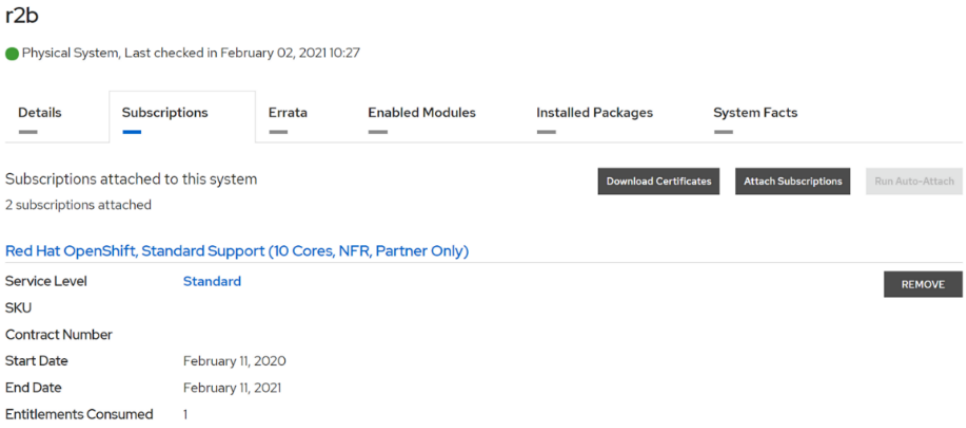

Figure 1. Downloading OpenShift subscription certificates

Extract the certificates file to the CSAH node

Run:

```
$ unzip cert_20210202.zip
$ ls
cert_20210202.zip  consumer_export.zip  openshift  signature
$ unzip consumer_export.zip
Archive:  consumer_export.zip
Candlepin export for beedde9c-a893-4a58-8313-4862e78806e5
  inflating: export/meta.json
  inflating:
export/entitlement_certificates/6834535004259762316.pem
```

Create the entitlement MachineConfig

1. Obtain the MachineConfig template by running the command:

   ```
   $ wget https://raw.githubusercontent.com/openshift-psap/blog-artifacts/master/how-to-use-entitled-builds-with-ubi/0003-cluster-wide-machineconfigs.yaml.template
   ```

2. Generate the MachineConfig file by appending the entitlement certificate:

   ```
   $ sed "s/BASE64_ENCODED_PEM_FILE/$(base64 -w 0
   </path/to/certificate_file.pem>)/g" 0003-cluster-wide-
   machineconfigs.yaml.template > 0003-cluster-wide-
   machineconfigs.yaml
   ```

3. Create the MachineConfig file, wait for the nodes to reboot, and then apply the configuration changes:

   ```
   $ oc create -f 0003-cluster-wide-machineconfigs.yaml
   ```

Validate the entitled builds

1. Download the validation pod yaml file and create the pod:

   ```
   wget https://raw.githubusercontent.com/openshift-psap/blog-artifacts/master/how-to-use-entitled-builds-with-ubi/0004-cluster-wide-entitled-pod.yaml
   oc create -f 0004-cluster-wide-entitled-pod.yaml
   ```

2. Examine the pod's logging output by running:

   ```
   oc logs cluster-entitled-build-pod
   ```

3. Validate that the pod can locate the various kernel-devel packages, as shown in the following figure:

```
Updating Subscription Management repositories.
Unable to read consumer identity
Subscription Manager is operating in container mode.
Red Hat Enterprise Linux 8 for x86_64 - BaseOS  9.1 MB/s |  27 MB     00:02
Red Hat Enterprise Linux 8 for x86_64 - AppStre 7.3 MB/s |  25 MB     00:03
Red Hat Universal Base Image 8 (RPMs) - BaseOS  2.0 MB/s | 772 kB     00:00
Red Hat Universal Base Image 8 (RPMs) - AppStre 6.2 MB/s | 4.9 MB     00:00
Red Hat Universal Base Image 8 (RPMs) - CodeRea 132 kB/s |  13 kB     00:00
Last metadata expiration check: 0:00:01 ago on Wed Feb  3 20:11:47 2021.
==================== Name Exactly Matched: kernel-devel ====================
kernel-devel-4.18.0-80.1.2.el8_0.x86_64 : Development package for building kernel modules to match the kernel
kernel-devel-4.18.0-80.el8.x86_64 : Development package for building kernel modules to match the kernel
kernel-devel-4.18.0-80.4.2.el8_0.x86_64 : Development package for building kernel modules to match the kernel
kernel-devel-4.18.0-80.7.1.el8_0.x86_64 : Development package for building kernel modules to match the kernel
kernel-devel-4.18.0-80.11.1.el8_0.x86_64 : Development package for building kernel modules to match the kernel
kernel-devel-4.18.0-147.el8.x86_64 : Development package for building kernel modules to match the kernel
kernel-devel-4.18.0-80.11.2.el8_0.x86_64 : Development package for building kernel modules to match the kernel
kernel-devel-4.18.0-80.7.2.el8_0.x86_64 : Development package for building kernel modules to match the kernel
kernel-devel-4.18.0-147.0.3.el8_1.x86_64 : Development package for building kernel modules to match the kernel
kernel-devel-4.18.0-147.8.1.el8_1.x86_64 : Development package for building kernel modules to match the kernel
kernel-devel-4.18.0-147.0.2.el8_1.x86_64 : Development package for building kernel modules to match the kernel
kernel-devel-4.18.0-147.3.1.el8_1.x86_64 : Development package for building kernel modules to match the kernel
kernel-devel-4.18.0-147.5.1.el8_1.x86_64 : Development package for building kernel modules to match the kernel
kernel-devel-4.18.0-193.el8.x86_64 : Development package for building kernel modules to match the kernel
kernel-devel-4.18.0-193.14.3.el8_2.x86_64 : Development package for building kernel modules to match the kernel
kernel-devel-4.18.0-193.13.2.el8_2.x86_64 : Development package for building kernel modules to match the kernel
kernel-devel-4.18.0-193.1.2.el8_2.x86_64 : Development package for building kernel modules to match the kernel
kernel-devel-4.18.0-193.19.1.el8_2.x86_64 : Development package for building kernel modules to match the kernel
kernel-devel-4.18.0-193.6.3.el8_2.x86_64 : Development package for building kernel modules to match the kernel
kernel-devel-4.18.0-240.el8.x86_64 : Development package for building kernel modules to match the kernel
kernel-devel-4.18.0-193.28.1.el8_2.x86_64 : Development package for building kernel modules to match the kernel
kernel-devel-4.18.0-240.1.1.el8_3.x86_64 : Development package for building kernel modules to match the kernel
kernel-devel-4.18.0-240.8.1.el8_3.x86_64 : Development package for building kernel modules to match the kernel
```

Figure 2. Locating the kernel-dev packages

Installing the NFD Operator

To install the NFD Operator, you must log in to the OpenShift cluster through the web console (see Accessing the OpenShift web console in the *Dell EMC Ready Stack for Red Hat OpenShift Container Platform 4.6 Deployment Guide*). Follow these steps:

1. Create a new project to manage the GPU resources:

   ```
   $ oc new-project gpu-operator-resources
   ```

2. From the web console, log into your OpenShift cluster, select **Operators** > **OperatorHub,** and then search for the **NFD Operator**.

   The Install Operator page opens, as shown in the following figure:

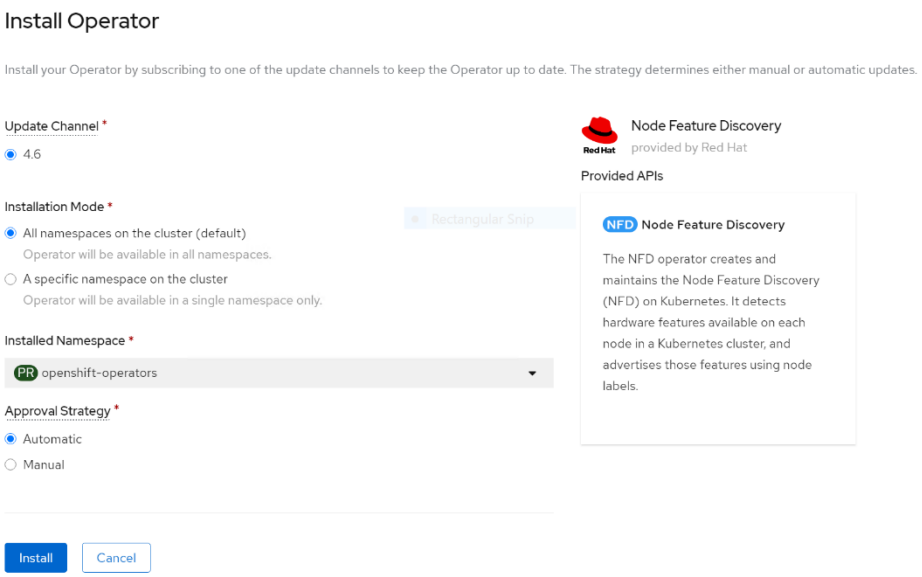

Figure 3. NFD Operator installation page

3. Choose to install the operator in **All namespaces on the cluster (default)** and select **Automatic** under **Approval Strategy**.
4. In the navigation pane, select **Operators** > **Installed Operators**, switch to the `gpu-operator-resources` namespace, and click **NFD Operator**.
5. Click the **Node Feature Discovery** tile, and then click **Create NodeFeatureDiscovery**.
6. Click **Create** to start the pods that are needed to label the nodes.

   After the NFD pods have started running, more compute node labels are added.
7. Validate that the GPU labels are present.

Note: Depending on the GPUs that you are using, the node label that the NFD generates might vary. The V100 GPUs in this example have the label `pci-de.present=true`.

Installing the NVIDIA GPU Operator

To install the NVIDIA GPU Operator:

1. Log in to the OpenShift web console, select **Operators** > **OperatorHub**, and then search for the **NVIDIA GPU Operator**.
2. Choose to install the Operator in **all namespaces in the cluster** and select the **automatic approval strategy**.
3. After the Operator is installed:
   a. Select **Operators** > **Installed Operators**, switch to the `gpu-operator-resources` project, and select the NVIDIA GPU Operator.
   b. Select the **ClusterPolicy** tab, click **Create ClusterPolicy**, and then click **Create** to start the NVIDIA GPU Operator pods.
   c. Under the `gpu-operator-resources` namespace, select **Workloads** > **Pods** and confirm that all NVIDIA GPU Operator pods are running or have completed, as shown in the following figure:

```
gpu-feature-discovery-5hh97                1/1     Running     0          8m9s    10.129.2.19    r2b-compute-2.r2b.oss.labs   <none>
        <none>
gpu-feature-discovery-6rb7x                1/1     Running     0          8m9s    10.131.0.22    r2b-compute-1.r2b.oss.labs   <none>
        <none>
gpu-feature-discovery-6w96p                1/1     Running     0          8m9s    10.128.2.27    r2b-compute-0.r2b.oss.labs   <none>
        <none>
nvidia-container-toolkit-daemonset-2stld   1/1     Running     0          10m     10.128.2.23    r2b-compute-0.r2b.oss.labs   <none>
        <none>
nvidia-container-toolkit-daemonset-9kkjh   1/1     Running     0          10m     10.131.0.18    r2b-compute-1.r2b.oss.labs   <none>
        <none>
nvidia-container-toolkit-daemonset-bbhbk   1/1     Running     0          10m     10.129.2.14    r2b-compute-2.r2b.oss.labs   <none>
        <none>
nvidia-dcgm-exporter-lfv4m                 1/1     Running     0          8m25s   10.131.0.21    r2b-compute-1.r2b.oss.labs   <none>
        <none>
nvidia-dcgm-exporter-qff5s                 1/1     Running     0          8m25s   10.129.2.18    r2b-compute-2.r2b.oss.labs   <none>
        <none>
nvidia-dcgm-exporter-wzvdj                 1/1     Running     0          8m25s   10.128.2.26    r2b-compute-0.r2b.oss.labs   <none>
        <none>
nvidia-device-plugin-daemonset-8bdpl       1/1     Running     0          8m51s   10.129.2.16    r2b-compute-2.r2b.oss.labs   <none>
        <none>
nvidia-device-plugin-daemonset-kkzxz       1/1     Running     0          8m51s   10.131.0.20    r2b-compute-1.r2b.oss.labs   <none>
        <none>
nvidia-device-plugin-daemonset-r4lbw       1/1     Running     0          8m51s   10.128.2.25    r2b-compute-0.r2b.oss.labs   <none>
        <none>
nvidia-device-plugin-validation            0/1     Completed   0          8m36s   10.129.2.17    r2b-compute-2.r2b.oss.labs   <none>
        <none>
nvidia-driver-daemonset-bp9d5              1/1     Running     0          11m     100.82.46.144  r2b-compute-0.r2b.oss.labs   <none>
        <none>
nvidia-driver-daemonset-c7x6q              1/1     Running     0          11m     100.82.46.145  r2b-compute-1.r2b.oss.labs   <none>
        <none>
nvidia-driver-daemonset-q8v9f              1/1     Running     0          11m     100.82.46.146  r2b-compute-2.r2b.oss.labs   <none>
        <none>
nvidia-driver-validation                   0/1     Completed   0          9m54s   10.128.2.24    r2b-compute-0.r2b.oss.labs   <none>
        <none>
```

Figure 4. NVIDIA GPU Operator pods status

A new `nvidia.com/gpu` resource is displayed In the NodeSpec for nodes with GPUs.

4. To confirm that the new resource is present, run:

   ```
   oc get node <gpu_node> -o yaml | grep -i nvidia.com/gpu
   ```

   The following output is displayed:

```
  nvidia.com/gpu.compute.major: "7"
  nvidia.com/gpu.compute.minor: "0"
  nvidia.com/gpu.count: "2"
  nvidia.com/gpu.family: volta
  nvidia.com/gpu.machine: PowerEdge-R740xd
  nvidia.com/gpu.memory: "32510"
  nvidia.com/gpu.present: "true"
  nvidia.com/gpu.product: Tesla-V100-PCIE-32
        f:nvidia.com/gpu.present: {}
        f:nvidia.com/gpu: {}
        f:nvidia.com/gpu: {}
        f:nvidia.com/gpu.compute.major: {}
        f:nvidia.com/gpu.compute.minor: {}
        f:nvidia.com/gpu.count: {}
        f:nvidia.com/gpu.family: {}
        f:nvidia.com/gpu.machine: {}
        f:nvidia.com/gpu.memory: {}
        f:nvidia.com/gpu.product: {}
  nvidia.com/gpu: "2"
  nvidia.com/gpu: "2"
```

Figure 5. Resources present

Providing GPU resources to a pod

Pod Spec

```
apiVersion: v1
kind: Pod
metadata:
 name: tensorflow-benchmarks-gpu
spec:
 nodeSelector:
   nvidia.com/gpu.product: Tesla-V100-PCIE-32GB
 containers:
 - image: nvcr.io/nvidia/tensorflow:19.09-py3
   name: cudnn
   command: ["/bin/sh","-c"]
   args: ["git clone
https://github.com/tensorflow/benchmarks.git;cd
benchmarks/scripts/tf_cnn_benchmarks;python3
tf_cnn_benchmarks.py --num_gpus=2 --data_format=NHWC --
batch_size=32 --model=resnet50 --
variable_update=parameter_server"]
   resources:
    limits:
      nvidia.com/gpu: 2
    requests:
      nvidia.com/gpu: 2
 restartPolicy: Never
```

As shown in the sample Pod Spec, you can provide GPUs to pods by specifying the GPU resource `nvidia.com/gpu` and requesting the number of GPUs that you want. This number must not exceed the number of GPUs present on a specific node.

The NVIDIA GPU Operator also deploys `gpu-feature-discovery` pods on each compute node. The pod labels each node with information about the GPU type, family, count, and so on, as shown in the Pod Spec. These node labels can be used in the Pod Spec to schedule workloads based on criteria such as the GPU product name, as shown under `nodeSelector`.

References

The following documentation provides additional information about the operation and usage of GPUs in the OpenShift environment:

- Red Hat OpenShift Container Platform Documentation
- NVIDIA GPUs on OpenShift Documentation