Hewlett Packard
Enterprise

Install Guide for Integration of HPE 3PAR Containerized Plugin with RedHat OpenShift / Kubernetes

Install Guide

Contents

Introduction

This document details the installation steps in order to get up and running quickly with the HPE 3PAR Volume Plug-in for Docker within a Kubernetes 1.7/Openshift 3.7 environment.

Before you begin:

- You need to have a **basic** knowledge of containers
- You should have Kubernetes or OpenShift deployed within your environment. If you want to learn how to deploy Kubernetes or OpenShift, please refer to the documents below for more details.

  Kubernetes

  https://kubernetes.io/docs/setup/independent/create-cluster-kubeadm/

  OpenShift

  https://docs.openshift.org/3.7/install_config/install/planning.html

Support Matrix for Kubernetes and Openshift 3.7

Platforms	Support for Containerized Plugin	Docker Engine Version	HPE 3PAR OS version
Kubernetes 1.6.13	Yes	1.12.6	3.2.2 MU6+ P107 3.3.1 MU1, MU2
Kubernetes 1.7.6	Yes	1.12.6	3.2.2 MU6+ P107 3.3.1 MU1, MU2
Kubernetes 1.8.9	Yes	17.06	3.2.2 MU6+ P107 3.3.1 MU1, MU2
Kubernetes 1.10.3	Yes	17.03	3.2.2 MU6+ P107 3.3.1 MU1, MU2
OpenShift 3.7 RPM based installation (Kubernetes 1.7.6)	Yes	1.12.6	3.2.2 MU6 + P107 3.3.1 MU1, MU2

Note:

- Managed Plugin is not supported for Kubernetes or Openshift 3.7
- The install of **OpenShift** for this paper was done on **RHEL 7.4**. Other versions of Linux may not be supported.

Deploying the HPE 3PAR Volume Plug-in for Docker as Docker Container (Containerized Plug-in) in Kubernetes/OpenShift

Below is the order and steps that will be followed to deploy the **HPE 3PAR Volume Plug-in for Docker (Containerized Plug-in)** within a **Kubernetes 1.7/OpenShift 3.7** environment.

Let's get started,

For this installation process, login in as **root**:

```
$ sudo su -
```

Configuring etcd

Note: For this quick start quide, we will be creating a single-node **etcd** deployment, as shown in the example below, but for production, it is **recommended** to deploy a High Availability **etcd** cluster.

For more information on **etcd** and how to setup an **etcd** cluster for High Availability, please refer: https://github.com/hpe-storage/python-hpedockerplugin/blob/master/docs/etcd_cluster_setup.md

1. **Export the Kubernetes/OpenShift Master node IP address**

```
$ export HostIP="<Master node IP>"
```

2. **Run the following to create the etcd container.**

NOTE: This **etcd** instance is separate from the **etcd** deployed by **Kubernetes/OpenShift** and is required for managing the **HPE 3PAR Volume Plug-in for Docker**. We need to modify the default ports (**2379, 4001, 2380**) of the new **etcd** instance to prevent conflicts. This allows two instances of **etcd** to safely run in the environment.

```
sudo docker run -d -v /usr/share/ca-certificates/:/etc/ssl/certs -p 40010:40010 \
-p 23800:23800 -p 23790:23790 \
--name etcd quay.io/coreos/etcd:v2.2.0 \
-name etcd0 \
-advertise-client-urls http://${HostIP}:23790,http://${HostIP}:40010 \
-listen-client-urls http://0.0.0.0:23790,http://0.0.0.0:40010 \
-initial-advertise-peer-urls http://${HostIP}:23800 \
-listen-peer-urls http://0.0.0.0:23800 \
-initial-cluster-token etcd-cluster-1 \
-initial-cluster etcd0=http://${HostIP}:23800 \
-initial-cluster-state new
```

Installing the HPE 3PAR Volume Plug-in for Docker (Containerized Plug-in) in Kubernetes or OpenShift:

NOTE: This section assumes that you already have **Kubernetes** or **OpenShift** deployed in your environment, in order to run the following commands.

1. **Install the iSCSI and Multipath packages**

```
yum install -y iscsi-initiator-utils device-mapper-multipath
```

2. **Configure /etc/multipath.conf**

```
$ vi /etc/multipath.conf
```

Copy the following into /etc/multipath.conf

```
defaults {
                polling_interval 10
                max_fds 8192
        }

        devices {
                device {
                        vendor                  "3PARdata"
                        product                 "VV"
                        no_path_retry           18
                        features                "0"
                        hardware_handler        "0"
                        path_grouping_policy    multibus
                        #getuid_callout         "/lib/udev/scsi_id --whitelisted --device=/dev/%n"
                        path_selector           "round-robin 0"
                        rr_weight               uniform
                        rr_min_io_rq            1
                        path_checker            tur
                        failback                immediate
                }
        }
```

Enable the **iscsid** and **multipathd** services

```
$ systemctl enable iscsid multipathd
$ systemctl start iscsid multipathd
```

3. **Configure MountFlags in the Docker service to allow shared access to Docker volumes**

```
$ vi /usr/lib/systemd/system/docker.service
Change MountFlags=slave to MountFlags=shared (default is slave)
```

Save and exit

4. **Restart the docker daemon**

```
$ systemctl daemon-reload
$ systemctl restart docker.service
```

5. **Setup the Docker plugin configuration file**

```
$ mkdir –p /etc/hpedockerplugin/
$ cd /etc/hpedockerplugin
$ vi hpe.conf
```

Copy the contents from the sample **hpe.conf** file, based on your storage configuration for either **iSCSI** or **Fiber Channel**:

HPE 3PAR iSCSI:
https://github.com/hpe-storage/python-hpedockerplugin/blob/plugin_v2/config/hpe.conf.sample.3parISCSI

```
host_etcd_ip_address = 192.x.x.x
host_etcd_port_number = 23790

#host_etcd_client_cert = /root/plugin/certs/<path to client certificate>.pem
#host_etcd_client_key = /root/plugin/certs/<path to client certificate key>.pem
```

HPE 3PAR Fiber Channel:
https://github.com/hpe-storage/python-hpedockerplugin/blob/plugin_v2/config/hpe.conf.sample.3parFC

6. **Deploy the HPE 3PAR Volume Plug-In for Docker (Containerized Plug-in) from the pre-built image available on Docker Hub:**

```
$ cd ~
$ vi docker-compose.yml
```

Copy the content below into the **docker-compose.yml** file:

```
hpedockerplugin:
  image: hpestorage/legacyvolumeplugin:2.1
  container_name: plugin_container
  net: host
  privileged: true
  volumes:
     - /dev:/dev
     - /run/lock:/run/lock
     - /var/lib:/var/lib
     - /var/run/docker/plugins:/var/run/docker/plugins:rw
     - /etc:/etc
     - /root/.ssh:/root/.ssh
     - /sys:/sys
     - /root/plugin/certs:/root/plugin/certs
     - /sbin/iscsiadm:/sbin/ia
     - /lib/modules:/lib/modules
     - /lib64:/lib64
     - /var/run/docker.sock:/var/run/docker.sock
     - /opt/hpe/data:/opt/hpe/data:rshared
```

Save and Exit

Note: Please make sure etcd service in running state.

7. **Start the HPE 3PAR Volume Plug-in for Docker (Containerized Plug-in)**

Make sure you are in the location of the **docker-compose.yml** file

```
$ docker-compose up -d or
$ docker-compose up 2>&1 | tee /tmp/plugin_logs.txt
```

Note: In case you are missing **docker-compose**, run the following commands:

```
$ curl -x -L https://github.com/docker/compose/releases/download/1.21.0/docker-compose-$(uname -s)-
$(uname -m) -o /usr/local/bin/docker-compose

$ sudo chmod +x /usr/local/bin/docker-compose
```

Vist https://docs.docker.com/compose/install/#install-compose for latest curl details

Test the installation

```
$ docker-compose --version
docker-compose version 1.21.0, build 1719ceb
```

Re-run the above commands.

For more information on Docker Compose, go to https://docs.docker.com/compose/install/#install-compose

8. **Success, you should now be able to test docker volume operations like:**

```
$ docker volume create -d hpe --name sample_vol -o size=1
```

9. **Install the HPE 3PAR FlexVolume driver:**

```
$ wget https://github.com/hpe-storage/python-hpedockerplugin/raw/master/dory_installer
$ chmod u+x ./dory_installer
$ sudo ./dory_installer
```

10. **Confirm HPE 3PAR FlexVolume driver installed correctly:**

```
$ ls -l /usr/libexec/kubernetes/kubelet-plugins/volume/exec/hpe.com~hpe/

-rwxr-xr-x. 1 docker docker 47046107 Apr 20 06:11 doryd
-rwxr-xr-x. 1 docker docker  6561963 Apr 20 06:11 hpe
-rw-r--r--. 1 docker docker      237 Apr 20 06:11 hpe.json
```

11. **Run the following command to start the HPE 3PAR FlexVolume dynamic provisioner:**

```
$ sudo /usr/libexec/kubernetes/kubelet-
plugins/volume/exec/hpe.com~hpe/doryd  /etc/kubernetes/admin.conf hpe.com
```

Note – If you see the following error:

```
Error getting config from file /etc/kubernetes/admin.conf - stat /etc/kubernetes/admin.conf: no such file or
directory
Error getting config cluster - unable to load in-cluster configuration, KUBERNETES_SERVICE_HOST and
KUBERNETES_SERVICE_PORT must be defined
```

Run the following commands:

```
$ mkdir –p /etc/kubernetes
$ cp /root/.kube/config /etc/kubernetes/admin.conf
```

Re-run the command to start the HPE 3PAR FlexVolume dynamic provisioner

For more information on the HPE FlexVolume driver, please visit this link:
https://github.com/hpe-storage/dory/blob/master/README.md

12. **Repeat steps 1-8 on all worker nodes. Step 9 only needs to be ran on the Master node.**

Upon successful completion of the above steps, you should have a working installation of Openshift 3.7 integrated with HPE 3PAR Volume Plug-in for Docker

Usage of the HPE 3PAR Volume Plug-in for Docker in Kubernetes/OpenShift

The following section will cover different operations and commands that can be used to familiarize yourself and verify the installation of the **HPE 3PAR Volume Plug-in for Docker** by provisioning storage using **Kubernetes/OpenShift** resources like **PersistentVolume**, **PersistentVolumeClaim**, **StorageClass**, **Pods**, etc.

To learn more about **Persistent Volume Storage** and Kubernetes/OpenShift, go to:
https://kubernetes.io/docs/tasks/configure-pod-container/configure-persistent-volume-storage/

Key Kubernetes/OpenShift Terms:

- **kubectl** – command line interface for running commands against Kubernetes clusters
- **oc** – command line interface for running commands against OpenShift platform
- **PV** – Persistent Volume is a piece of storage in the cluster that has been provisioned by an administrator.
- **PVC** – Persistent Volume Claim is a request for storage by a user.
- **SC** – Storage Class provides a way for administrators to describe the “classes” of storage they offer.
- **hostPath volume** – mounts a file or directory from the host node’s filesystem into your Pod.

To get started, in an OpenShift environment, we need to relax the security of your cluster so pods are allowed to use the **hostPath** volume plugin without granting everyone access to the privileged **SCC**:

1. Edit the **restricted** SCC:
```
$ oc edit scc restricted
```
2. Add **allowHostDirVolumePlugin: true**
3. Save the changes
4. Restart node service (master node).
```
$ sudo systemctl restart origin-node.service
```

Below is an example **yaml** specification to create **PersistentVolume** (PV) using the **HPE 3PAR FlexVolume driver**.

Note: If you have OpenShift installed, **kubectl create** and **oc create** commands can be used interchangeably when creating PVs, PVCs, and SCs.

Dynamic volume provisioning allows storage volumes to be created on-demand. To enable dynamic provisioning, a cluster administrator needs to pre-create one or more **StorageClass** objects for users. **StorageClass** object defines the **storage provisioner** and parameters to be used during dynamic storage provisioning requests made within a **Kubernetes/Openshift** environment. This provisioner is a simple daemon that listens for **PVCs** and satisfies those claims based on the defined **StorageClass** .

The following creates a **StorageClass** **sc1** which provisions a compressed volume with the help of **HPE 3PAR Docker Volume Plugin**.

```
$ sudo kubectl create -f - << EOF
---
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
 name: sc1
provisioner: hpe.com/hpe
parameters:
  size: "16"
  compression: "true"
EOF
```

Note : Compression : True supported on 3PAR OS 3.3.1 MU1 onwards.

Now let’s create a claim **PersistentVolumeClaim** (PVC):

```
$ sudo kubectl create -f - << EOF
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
  storageClassName: sc1
EOF
```

Looking at the **yaml** file, under **flexVolume** spec and in the **driver** attribute, we specify the **hpe** FlexVolume driver to be used. We can also specify the **docker volume provisioner** specs (**name**, **size**, **compression**, etc.) under **options**.

```
$ sudo kubectl create -f - << EOF
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv1
spec:
    capacity:
      storage: 20Gi
    accessModes:
    - ReadWriteOnce
    flexVolume:
      driver: hpe.com/hpe
      options:
        name: hpe_volume
        size: "20"
EOF
```

At this point after creating the **SC** and **PVC** definitions, the volume hasn't been created yet. The actual volume gets created on-the-fly during the pod deployment and volume mount phase.

So let's create a **pod** using the **nginx** container along with some persistent storage:

```
$ sudo kubectl create -f - << EOF
---
apiVersion: v1
kind: Pod
metadata:
  name: pod1
spec:
  containers:
  - name: nginx
    image: nginx
    volumeMounts:
    - name: export
      mountPath: /export
  restartPolicy: Always
  volumes:
  - name: export
    persistentVolumeClaim:
      claimName: pvc1
EOF
```

When the **pod** gets created and a mount request is made, the volume is now available and can be seen using the following command:

```
$ docker volume ls
DRIVER              VOLUME NAME
hpe                 export
```

On the Kubernetes/Openshift side it should now look something like this:

```
$ kubectl get pv,pvc,pod -o wide
NAME       CAPACITY   ACCESSMODES   RECLAIMPOLICY   STATUS    CLAIM           STORAGECLASS   REASON   AGE
pv/pv1     20Gi       RWO           Retain          Bound     default/pvc1                            11m

NAME         STATUS    VOLUME    CAPACITY   ACCESSMODES   STORAGECLASS   AGE
pvc/pvc1     Bound     pv100     20Gi       RWO                          11m

NAME                        READY     STATUS    RESTARTS   AGE       IP             NODE
po/pod1                     1/1       Running   0          11m       10.128.1.53    cld6b16

Run : oc get pod pod1
[root@cssosbe04-b01 ~]# oc get pod pod1
NAME      READY     STATUS    RESTARTS   AGE
pod1      1/1       Running   0          4moc
```

Now the **pod** can be deleted to **unmount** the **docker volume**. Deleting a **docker volume** does not require manual clean-up because the **dynamic provisioner** provides automatic clean-up. You can delete the **PersistentVolumeClaim** and see the **PersistentVolume** and **docker volume** automatically deleted.

Congratulations, you have completed all validation steps and have a working Kubernetes/OpenShift environment.

Appendix

Restarting the plugin

```
$ docker stop <container_id_of_plugin>
```

IMPORTANT NOTE: The /run/docker/plugins/hpe.sock and /run/docker/plugins/hpe.sock.lock files are not automatically removed when you stop the container. Therefore these files will need to be removed manually between each run of the plugin.

```
$ docker start <container_id_of_plugin>
```

Usage

For more details on HPE 3PAR Volume Plugin-in for Docker usage and its features refer: https://github.com/hpe-storage/python-hpedockerplugin/blob/master/docs/usage.md

Known Issues:

All the known issues regarding plugin can be found at the link below:https://github.com/hpe-storage/python-hpedockerplugin/issues

Learn more visit

https://www.hpe.com/storage
https://developer.hpe.com/platform/hpe-3par/home

4AA4-xxxxENW, Month 20XX

Hewlett Packard Enterprise