DockerEE Pure Storage Install

Notes:

- The Kubernetes plugin depends on the `purestorage/docker-plugin` this is installed automatically but reuires it's own dependencies.
- Latest linux multipath software package for your operating system (Required)
- Latest iSCSI initiator software for your operating system (Optional, required for iSCSI connectivity)
- Latest NFS software package for your operating system (Optional, required for NFS connectivity)
- Latest FC initiator software for your operating system (Optional, requied for FC connectivity)
- Latest Filesystem utilities/drivers (XFS by default, Required)

References:

- https://blog.2vcps.io/2018/03/08/kubernetes-and-the-pure-storage-flexvolume-plugin/
- https://github.com/purestorage/helm-charts/tree/master/pure-k8s-plugin
- https://support.purestorage.com/Solutions/Kubernetes/Kubernetes%2C_Persistent_Volumes%2C_and_Pure_Storage
- https://hub.docker.com/r/purestorage/docker-plugin

Step 0:

- Ensure any machine that mounts a `Pure` volume has the reuired drivers mentioned in the above notes.
- Ensure you have the correct context for your cluster `kubectl get nodes`

Step 1: Install Helm

1. Download binary release
2. `chmod +x $file && mv $file /usr/bin/helm`
3. `helm init` (this will install tiller on the Kubernetes cluster)

Step 2: Add Pure Storage helm repo (cloning to your own repository and adding is also an option)

```
helm repo add pure https://purestorage.github.io/helm-charts
helm repo update
helm search pure-k8s-plugin
```

Step 3: Configure Service Accounts

1. `kubectl create clusterrolebinding add-on-cluster-admin --clusterrole=cluster-admin --serviceaccount=${TILLER_NAMESPACE}:default`

Step 4: Confiigure the Helm configuration.

1. Copy the `values.yml` from https://github.com/purestorage/helm-charts/blob/master/pure-k8s-plugin/values.yaml into a local dir`

2. Modify the `values.yml` these are the configurable values in the chart that will template the deployment. Below are some snippets of imprtant configuration vaules that may reuire additinal support from Pure or the storage team. They are included in this document for discussion purposes. See https://github.com/purestorage/helm-charts/tree/master/pure-k8s-plugin#configuration for detailed info about these config values.

```
arrays:
#FlashArrays:
#  - MgmtEndPoint: "1.2.3.4"
#    APIToken: "a526a4c6-18b0-a8c9-1afa-3499293574bb"
#  #Labels can be used to schedule pods on certain nodes.  This can help keep workloads near the storage
#    Labels:
#      rack: "22"
#      env: "prod"
#  - MgmtEndPoint: "1.2.3.5"
#    APIToken: "b526a4c6-18b0-a8c9-1afa-3499293574bb"
#FlashBlades:
#  - MgmtEndPoint: "1.2.3.6"
#    APIToken: "T-c4925090-c9bf-4033-8537-d24ee5669135"
#    NfsEndPoint: "1.2.3.7"
#    Labels:
#      rack: "7b"
#      env: "dev"
#  - MgmtEndPoint: "1.2.3.8"
#    APIToken: "T-d4925090-c9bf-4033-8537-d24ee5669135"
#    NfsEndPoint: "1.2.3.9"
#    Labels:
#      rack: "6a"
```

```
    # support ISCSI or FC, not case sensitive
flasharray:
    sanType: ISCSI
    defaultMountOpt: ""
    preemptAttachments: "true"
```

Step 5: Install the plugin

1. Do a dry run

`helm install --name pure-storage-driver pure/pure-k8s-plugin --namespace pure -f <your_own_dir>/yourvalues.yaml --dry-run --debug`

2. Install the plugin

```
helm install --name pure-storage-driver pure/pure-k8s-plugin --namespace pure -f <your_own_dir>/yourvalues.yaml
```

Step 6: Test

Enter the following, you should get the output shown

```
kubectl get sc


NAME TYPE
pure pure-provisioner
pure-block pure-provisioner
pure-file pure-provisioner
```

Create the following objects with `kubectl create -f`

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: pure-block
  labels:
  kubernetes.io/cluster-service: “true”
provisioner: pure-provisioner
parameters:
  backend: block
```

```
kind: PersistentVolume
apiVersion: v1
metadata:
  name: pure-pv-volume
spec:
  storageClassName: pure-block
  capacity:
  storage: 10Gi
  accessModes:
  - ReadWriteOnce
 hostPath:
 path: “/tmp/data”
```

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pure-volume
```

```
spec:
  accessModes:
  - ReadWriteOnce
 resources:
 requests:
 storage: 10Gi
 storageClassName: pure-block
```

Now ensure that the Claim is bound `kubectl get pvc`

At this point you a are able to provision storage for workloads using the pure storage k8s plugin.