Local Storage

HostPath Volume

Simply add hostpath to volumes section in the deployment definition:

```
     volumes:
     - name: local-vol
       hostPath:
         path: {{ .Values.persistentVolume.path }}
         type: DirectoryOrCreate
```

Example:

```
kind: Deployment
...
    spec:
      containers:
      - name: registry
...
        volumeMounts:
        - mountPath: /var/lib/registry
          name: local-vol
          subPath: registry/data
      volumes:
      - name: local-vol
        hostPath:
          path: {{ .Values.persistentVolume.path }}
          type: DirectoryOrCreate
...
```

HostPath Storage using Persistent Volume

This is the simplest and best approach for bare metal deployments when a network file system is not available.

Define storage class and make it the default

helm-template.yaml

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: hostpath-storage
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: keystone/hostpath-storage
volumeBindingMode: Immediate
reclaimPolicy: Retain
```

Remove default status from other storage classes

> kubectl patch storageclass <STORAGE_CLASS> -p '{"metadata": {"annotations":{"storageclass.kubernetes.io/is-default-class":"false"}}}'

Create Persistent Volumes

This example creates 10 persistent volumes using helm

helm-template.yaml

```
{{- $root := . -}}
{{range $i, $e := until 10}}
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-{{ $i }}
spec:
  capacity:
    storage: {{ $root.Values.persistentVolume.size }}
  volumeMode: Filesystem
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: hostpath-storage
  hostPath:
    path: /var/pv/pv-{{ $i }}
    type: DirectoryOrCreate
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: NotIn
          values:
          - master
---
{{end}}
```

Make a Claim using Default Storage Class

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: my-claim
spec:
  storageClassName: ""
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
```

Local Storage using Persistent Volume and Claim

We can use disk space on a node by defining a PersistentVolume (see below) and then making a claim against that volume by specifying the storageclass name in the PersistentVolumeClaim.

- Only one claim can be made against a volume.
- File path (local.path) must exist for the volume to be usable.
- USE HOSTPATH STORAGE since it will create the folders for you.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: local-storage
spec:
  capacity:
    storage: 10Gi
  # volumeMode field requires BlockVolume Alpha feature gate to be enabled.
  volumeMode: Filesystem
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Persist
  storageClassName: local-storage
  local:
    path: /var/k8s/LOCAL_STORAGE
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - k8sworker1
          - k8sworker2
          - k8sworker3
          - docker-for-desktop
```

Make a claim by specifying the storage class

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: local-storage-claim
spec:
  storageClassName: local-storage
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 3Gi
```

Make any required folders on the worker nodes:

> *ssh k8sworker1*
>
> *sudo mkdir -p /var/k8s/LOCAL_STORAGE*

Repeat for all nodes requiring local storage.

Using Default Storage Class with Prebuilt Persistent Volumes

We can create a storage class for our local-storage and use it as default storage. The only issue with doing this with local-storage is that we need to pre-build all of the persistent volumes. Since only 1 claim can be made against a volume, we will need to make a few.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: local-storage
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: keystone/local-storage
volumeBindingMode: Immediate
reclaimPolicy: Retain
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: local-storage-1
spec:
  capacity:
    storage: 2Gi
  volumeMode: Filesystem
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: local-storage
  local:
    path: /var/pv1
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: NotIn
          values:
          - master
---
...REPEAT UNTIL HAPPY ...
```

To use the default storage of the cluster, you just need to create a claim and specify "" for storageClassName.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: local-storage-claim
spec:
  storageClassName: ""
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 3Gi
```

GlusterFS

Gluster-kubernetes is a project to provide Kubernetes administrators a mechanism to easily deploy GlusterFS as a native storage service onto an existing Kubernetes cluster.

See https://github.com/gluster/gluster-kubernetes

References

Reference	URL
Volumes - Kubernetes	https://kubernetes.io/docs/concepts/storage/volumes/
Local Persistent Volumes for Kubernetes Goes Beta	https://kubernetes.io/blog/2018/04/13/local-persistent-volumes-beta/
Change Default Storage Class	https://kubernetes.io/docs/tasks/administer-cluster/change-default-storage-class/
Bare Metal Storage	https://medium.com/devityoself/kubernetes-bare-metal-storage-49b69d090dfa
GlusterFS Native Storage Service for Kubernetes	https://github.com/gluster/gluster-kubernetes