How do I create a storage class for NFS dynamic storage provisioning in an OpenShift environment?

How To

Summary

When you create a demo environment for the Cloud Pak, it requires a storage class to dynamically provision persistent volumes connected to storage with enough space for the environment. If you don't already have one available, these steps can be used to set up an open source NFS dynamic storage provisioner on OpenShift.

Environment

These steps can be executed from any client machine with an `oc` Command Line Interface (CLI).

It requires that you have an NFS share already set-up with the needed space. Your OpenShift cluster needs to be able to access the NFS share.

Your OpenShift user needs to have access to the project where you deploy the provisioner. When you use the "default' namespace, you would need to be a cluster administrator. The oc adm command also requires administrator rights.

This document is written based on v4.0.2 instructions. It is possible for some things to change over time and it is good to review the associated instructions.

Steps

These steps summarize the procedure needed to set up the NFS-Client dynamic storage provisioner. This third-party provisioner is available and documented at Kubernetes NFS Subdir External Provisioner. *(https://github.com/kubernetes-sigs/nfs-subdir-external-provisioner)*
Update: The without Helm instructions for this storage class improved significantly since I first wrote this article. You might want to follow the instructions that come with the provisioner.

1. Ensure you have an NFS share setup and accessible to your OpenShift cluster. You need to know the following details:
   - NFS server hostname or IP address
   - The path for your shared NFS directory
2. Download the setup files for the provisioner from GitHub at Kubernetes NFS Subdir External Provisioner. *(https://github.com/kubernetes-sigs/nfs-subdir-external-provisioner)*.
3. Extract the files and find the needed yaml files at `nfs-subdir-external-provisioner-master\deploy\`.
4. Run these two commands to deploy the provisioner security settings:
   ```
   oc create -f rbac.yaml
   oc adm policy add-scc-to-user hostmount-anyuid
   system:serviceaccount:default:nfs-client-provisioner
   ```
   Note: If you need to deploy this to a project other than "default", replace each mention of `default` for the namespace in rbac.yaml, deployment.yaml and the oc adm command.

5. Edit the deployment.yaml file to include the proper NFS host and path for each location it is used. For example:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nfs-client-provisioner
  labels:
    app: nfs-client-provisioner
  # replace with namespace where provisioner is deployed
  namespace: default
spec:
  replicas: 1
  strategy:
    type: Recreate
  selector:
    matchLabels:
      app: nfs-client-provisioner
  template:
    metadata:
      labels:
        app: nfs-client-provisioner
    spec:
      serviceAccountName: nfs-client-provisioner
      containers:
        - name: nfs-client-provisioner
          image: k8s.gcr.io/sig-storage/nfs-subdir-external-provisioner:v4.0.2
          volumeMounts:
            - name: nfs-client-root
              mountPath: /persistentvolumes
          env:
            - name: PROVISIONER_NAME
              value: k8s-sigs.io/nfs-subdir-external-provisioner
            - name: NFS_SERVER
              value: MyNFSHostname
            - name: NFS_PATH
              value: /nfs/cpshare/
      volumes:
        - name: nfs-client-root
          nfs:
            server: MyNFSHostname
            path: /nfs/cpshare/
```

6. Run this command to deploy the NFS-client provisioner.
   `oc create -f deployment.yaml`
   Note: The container for this provisioner is coming from k8s.gcr.io.
7. Edit the class.yaml file to use the wanted names. In this example the storage class name is managed-nfs-storage.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: managed-nfs-storage
provisioner: k8s-sigs.io/nfs-subdir-external-provisioner # or choose another name
parameters:
  pathPattern: "${.PVC.namespace}-${.PVC.name}"
  archiveOnDelete: "false"
```

8. Create the storage class for your dynamic provisioner.
   `oc create -f class.yaml`
9. As mentioned in the documentation, you can also test it out with these steps:
   a. Create a test PVC and pod with this command:
      `oc create -f test-claim.yaml -f test-pod.yaml`
      Note: `test-pod.yaml points to` busybox:stable, your environment might not be able to resolve that. You might have to pull the busybox image into your image registry from docker hub.
   b. Check for SUCCESS file created in your NFS share directory at the path for the PVC.
   c. Clean up the config and pods with this command:
      `oc delete -f test-pod.yaml -f test-claim.yaml`

You now have a storage class that can be used to dynamically provision persistent volumes on your NFS share.

Document Information

More support for:

IBM Cloud Pak for Automation *(https://www.ibm.com/mysupport/s/topic/0TO0z000000YgPmGAK)*

Component:

CloudPak4Automation Platform

Software version:

All Version(s)

Document number:

6356629

Modified date:

26 January 2022