文档名称：ZDNS-Zcloud-kubernetes 网络存储-持久化-动态分配-NFS-实施方案
文档分类：ZDNS-Zcloud-kubernetes 文档
服务等级：
文档编号：
作　　者: 余春云
联系电话：18811483030
电子邮件：yuchunyun@zdns.cn

Kubernetes 网络存储-持久化-动态分配-NFS-实施方案

文档变更历史记录：

变更日期	变更人	变更内容摘要	组长确认
2019-03-14	余春云	初次建立文档	

目录

Kubernetes 网络存储-持久化-动态分配-NFS-实施方案.....1
目的.....3
说明.....3
配置步骤.....3
RBAC 配置.....3
nfs-provisioner 配置.....4
StorageClass 配置.....6
创建 pod 通过 pvc 使用 storageclass 来申请自动分配 PV.....7

目的

允许用户通过标准 PVC 接口以简单、便携的方式访问网络存储 NFS

说明

nfs-provisioner 部署成 StatefulSet 形式的 pod，该 pod 集成了 NFS 服务器和 provisioner，因此需要集群中提前配置好可用的静态 PV 或者可用的 storageclass 来动态创建 PV（本方案采用后者）。

配置步骤

RBAC 配置

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: nfs-provisioner
---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: nfs-provisioner-runner
rules:
  - apiGroups: [""]
    resources: ["persistentvolumes"]
    verbs: ["get", "list", "watch", "create", "delete"]
  - apiGroups: [""]
    resources: ["persistentvolumeclaims"]
    verbs: ["get", "list", "watch", "update"]
  - apiGroups: ["storage.k8s.io"]
    resources: ["storageclasses"]
    verbs: ["get", "list", "watch"]
  - apiGroups: [""]
    resources: ["events"]
    verbs: ["create", "update", "patch"]
  - apiGroups: [""]
    resources: ["services", "endpoints"]
    verbs: ["get"]
  - apiGroups: ["extensions"]
    resources: ["podsecuritypolicies"]
```

```
    resourceNames: ["nfs-provisioner"]
    verbs: ["use"]
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: run-nfs-provisioner
subjects:
  - kind: ServiceAccount
    name: nfs-provisioner
      # replace with namespace where provisioner is deployed
    namespace: default
roleRef:
  kind: ClusterRole
  name: nfs-provisioner-runner
  apiGroup: rbac.authorization.k8s.io
---
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: leader-locking-nfs-provisioner
rules:
  - apiGroups: [""]
    resources: ["endpoints"]
    verbs: ["get", "list", "watch", "create", "update", "patch"]
---
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: leader-locking-nfs-provisioner
subjects:
  - kind: ServiceAccount
    name: nfs-provisioner
    # replace with namespace where provisioner is deployed
    namespace: default
roleRef:
  kind: Role
  name: leader-locking-nfs-provisioner
  apiGroup: rbac.authorization.k8s.io
```

nfs-provisioner 配置

```
kind: Service
apiVersion: v1
```

```
metadata:
  name: nfs-provisioner
  labels:
    app: nfs-provisioner
spec:
  ports:
    - name: nfs
      port: 2049
    - name: mountd
      port: 20048
    - name: rpcbind
      port: 111
    - name: rpcbind-udp
      port: 111
      protocol: UDP
  selector:
    app: nfs-provisioner
---
kind: StatefulSet
apiVersion: apps/v1beta2
metadata:
  name: nfs-provisioner
spec:
  selector:
    matchLabels:
      app: nfs-provisioner
  replicas: 1
  serviceName: nfs-provisioner
  template:
    metadata:
      labels:
        app: nfs-provisioner
    spec:
      serviceAccount: nfs-provisioner
      containers:
        - name: nfs-provisioner
          image: quay.io/kubernetes_incubator/nfs-provisioner:latest
          ports:
            - name: nfs
              containerPort: 2049
            - name: mountd
              containerPort: 20048
            - name: rpcbind
              containerPort: 111
```

```
                - name: rpcbind-udp
                  containerPort: 111
                  protocol: UDP
              securityContext:
                capabilities:
                  add:
                    - DAC_READ_SEARCH
                    - SYS_RESOURCE
              args:
                - "-provisioner=example.com/nfs"
              env:
                - name: POD_IP
                  valueFrom:
                    fieldRef:
                      fieldPath: status.podIP
                - name: SERVICE_NAME
                  value: nfs-provisioner
                - name: POD_NAMESPACE
                  valueFrom:
                    fieldRef:
                      fieldPath: metadata.namespace
              imagePullPolicy: "IfNotPresent"
              volumeMounts:
                - name: nfs-data
                  mountPath: /export
  volumeClaimTemplates:
    - metadata:
        name: nfs-data
      spec:
        storageClassName: “local-volume-dynamic-file "
        accessModes: [ "ReadWriteOnce" ]
        resources:
          requests:
            storage: "7Gi"
```

注：需要集群中存在一个可用 storageclass，用来分配一个 PV 供 NFS 服务器使用

StorageClass 配置

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: example-nfs
provisioner: example.com/nfs
mountOptions:
```

```
  - vers=4.1
```

注：通过该 storageclass 自动创建的 pv 回收策略默认是 Delete，如果想换成或者增加一个回收策略是 Retain，需要手动指定 storageclass 的 reclaimPolicy: Retain

可以通过 StorageClass.metadata.annotations，将某个 storageclass 设置为默认的存储类

```
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
```

创建 pod 通过 pvc 使用 storageclass 来申请自动分配 PV

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: nfs-pv-claim-1
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: "example-nfs"
  resources:
    requests:
      storage: 1Gi
---
kind: Pod
apiVersion: v1
metadata:
  name: mypod
spec:
  containers:
    - name: client
      image: ikubernetes/myapp:v3
      volumeMounts:
      - mountPath: "/data"
        name: data
  volumes:
    - name: data
      persistentVolumeClaim:
        claimName: nfs-pv-claim-1
```