kubernetes搭建高可用nacos

建立日期：2020年8月13日

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 版本 | 变更原因 | 变更内容简述 | 编制/修订者 | 适用环境 | 发布日期 |
| V1.0 | 建立 |  | 陈文华 | 开发、测试 |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

## 一、部署NFS-Client Provisioner、NFS StorageClass

在Kubernetes上部署Nacos的官网文档地址

<https://nacos.io/zh-cn/docs/use-nacos-with-kubernetes.html>

1、在master服务器上创建nfs目录用来存放资源配置文件

mkdir /home/nfs

2、进入nfs目录，创建角色资源文件

cd /home/nfs

vim rbac.yml

内容如下所示：

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

3、创建角色

kubectl apply -f rbac.yml

4、在nfs服务器(node1节点84.48)创建/nfs/nfs-share目录

mkdir /nfs/nfs-share

5、创建NFS-Client Provisioner资源文件

vim client-provisioner-deployment.yml

内容如下：

**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  
 **selector:  
 matchLabels:  
 app:** nfs-client-provisioner  
 **strategy:  
 type:** Recreate  
 **template:  
 metadata:  
 labels:  
 app:** nfs-client-provisioner  
 **spec:  
 serviceAccountName:** nfs-client-provisioner  
 **containers:** - **name:** nfs-client-provisioner  
 **image:** quay.io/external\_storage/nfs-client-provisioner:latest  
 **volumeMounts:** - **name:** nfs-client-root  
 **mountPath:** /persistentvolumes  
 **env:** - **name:** PROVISIONER\_NAME  
 **value:** cloudkube.com/nfs *# 名称可自定义* - **name:** NFS\_SERVER  
 **value:** 192.168.84.48 *# NFS服务器IP* - **name:** NFS\_PATH  
 **value:** /nfs/nfs-share *# NFS目录* **volumes:** - **name:** nfs-client-root  
 **nfs:  
 server:** 192.168.84.48 *# NFS服务器IP* **path:** /nfs/nfs-share *# NFS目录*

6、部署 NFS-Client Provisioner

kubectl apply -f client-provisioner-deployment.yml

7、创建 NFS StorageClass资源

vim storageclass.yml

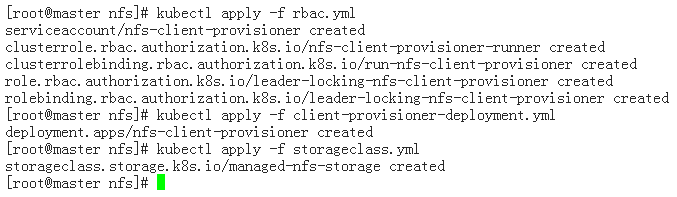
内容如下：

**apiVersion:** storage.k8s.io/v1  
**kind:** StorageClass  
**metadata:  
 name:** managed-nfs-storage *# StorageClass名称，可自定义***provisioner:** cloudkube.com/nfs *# 和client-provisioner-deployment.yml里定义的一致***parameters:  
 archiveOnDelete: "false"**

8、创建 NFS StorageClass

kubectl apply -f storageclass.yml

以上资源创建过程



## 二、部署Nacos主数据库

1、master服务器创建nacos目录，存放资源配置文件

mkdir /home/nacos

2、创建ConfigMap配置文件，定义部署Nacos集群需要用到的所有配置

cd /home/nacos

vim nacos-mysql-configmap.yml

内容如下：

**apiVersion:** v1  
**kind:** ConfigMap  
**metadata:  
 name:** nacos-mysql-cm  
**data:  
 nacos.mysql.master.root.password: "nacos\_root"** *# 主数据库root密码* **nacos.mysql.master.db: "platform\_nacos"** *# 主数据库库名* **nacos.mysql.master.user: "nacos"** *# 主数据库用户名* **nacos.mysql.master.password: "nacos"** *# 主数据库密码* **nacos.mysql.master.port: "3306"** *# 主数据库端口* **nacos.mysql.replication.user: "nacos\_rep"** *# 拷贝用的账户* **nacos.mysql.replication.password: "nacos\_rep"** *# 拷贝用的密码* **nacos.mysql.slave.root.password: "nacos\_root"** *# 从数据库root密码* **nacos.mysql.slave.port: "3306"** *# 从数据库端口*

3、创建ConfigMap

kubectl apply -f nacos-mysql-configmap.yml

4、登录NFS服务器（node1节点84.48），创建主从库NFS挂载目录

mkdir /nfs/nacos-mysql-master

mkdir /nfs/nacos-mysql-slave

5、创建主数据库的资源配置文件

vim nacos-mysql-master.yml

内容如下：

**apiVersion:** v1  
**kind:** ReplicationController  
**metadata:  
 name:** nacos-mysql-master  
 **labels:  
 name:** nacos-mysql-master  
**spec:  
 replicas:** 1  
 **selector:  
 name:** nacos-mysql-master  
 **template:  
 metadata:  
 labels:  
 name:** nacos-mysql-master  
 **spec:  
 containers:** - **name:** master  
 **image:** nacos/nacos-mysql-master:latest *# nacos-mysql-master镜像* **ports:** - **containerPort:** 3306 *# 端口3306* **volumeMounts:** - **mountPath:** /var/lib/mysql  
 **name:** nacos-mysql-master-data  
 *# 从nacos-mysql-cm中读取配置，加载到环境变量* **env:** - **name:** MYSQL\_ROOT\_PASSWORD  
 **valueFrom:  
 configMapKeyRef:  
 name:** nacos-mysql-cm  
 **key:** nacos.mysql.master.root.password  
 - **name:** MYSQL\_DATABASE  
 **valueFrom:  
 configMapKeyRef:  
 name:** nacos-mysql-cm  
 **key:** nacos.mysql.master.db  
 - **name:** MYSQL\_USER  
 **valueFrom:  
 configMapKeyRef:  
 name:** nacos-mysql-cm  
 **key:** nacos.mysql.master.user  
 - **name:** MYSQL\_PASSWORD  
 **valueFrom:  
 configMapKeyRef:  
 name:** nacos-mysql-cm  
 **key:** nacos.mysql.master.password  
 - **name:** MYSQL\_REPLICATION\_USER  
 **valueFrom:  
 configMapKeyRef:  
 name:** nacos-mysql-cm  
 **key:** nacos.mysql.replication.user  
 - **name:** MYSQL\_REPLICATION\_PASSWORD  
 **valueFrom:  
 configMapKeyRef:  
 name:** nacos-mysql-cm  
 **key:** nacos.mysql.replication.password  
 **volumes:** *# 指定nfs服务挂载* - **name:** nacos-mysql-master-data  
 **nfs:  
 path:** /nfs/nacos-mysql-master  
 **server:** 192.168.84.48  
---  
*# 创建nacos主库service，供nacos server连接***apiVersion:** v1  
**kind:** Service  
**metadata:  
 name:** mysql-master *# 必须叫这个名字，因为nacos server内部根据这个名字连接数据库* **labels:  
 name:** mysql-master  
**spec:  
 ports:** - **port:** 3306  
 **targetPort:** 3306  
 **selector:  
 name:** nacos-mysql-master

6、创建资源

kubectl apply -f nacos-mysql-master.yml

7、查看pod创建过程

kubectl get pods -w | grep nacos

如果出现ImagePullBackOff错误

vim /usr/lib/systemd/system/kubelet.service.d/10-kubeadm.conf

添加

Environment="KUBELET\_OPTS=--image-pull-progress-deadline=60m"

最后一行ExecStart添加$KUBELET\_OPTS重启kubelet

systemctl stop kubelet

systemctl daemon-reload

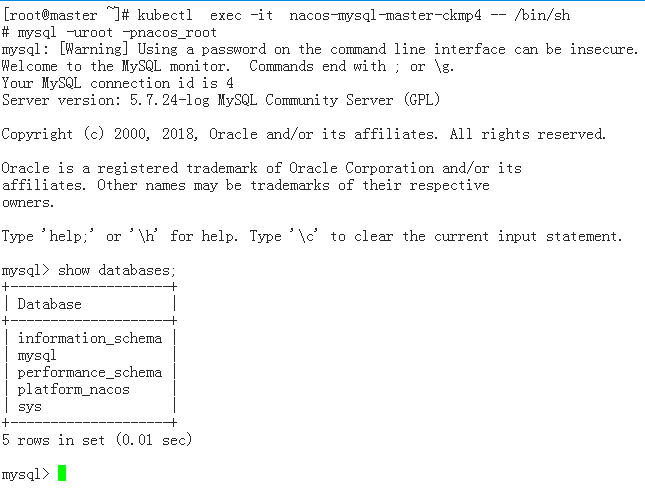
systemctl start kubelet

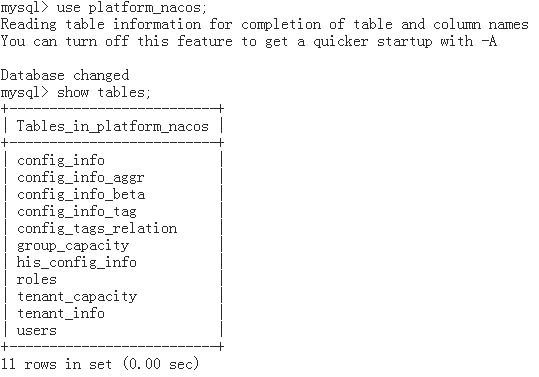
如果还是报此错误，master服务器下载镜像推送到Harbor，worker节点从Harbor下载镜像，docker tag修改镜像名称，或者nacos-mysql-master.yml 文件修改成本地仓库的镜像名



8、进入容器内部登录mysql

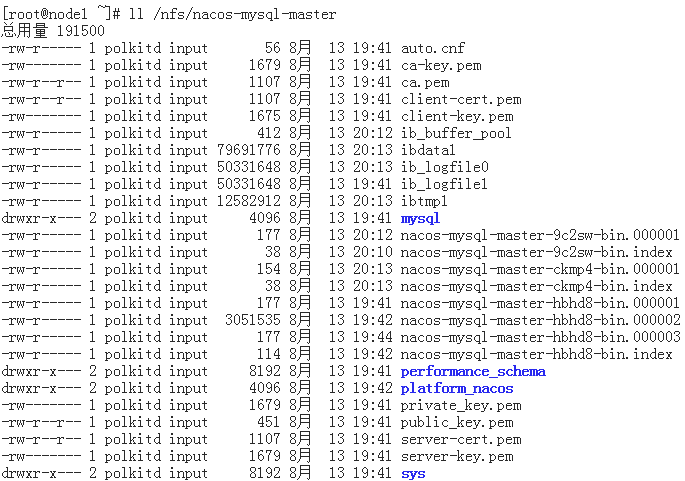
用户名和密码就是ConfigMap中配置的





查看node1节点nfs服务器目录下文件

ll /nfs/nacos-mysql-master



查看SVC，已绑定Endpoins



## 三、部署Nacos从数据库

1、创建从数据库资源文件

vim nacos-mysql-slave.yml

内容如下：

**apiVersion:** v1  
**kind:** ReplicationController  
**metadata:  
 name:** nacos-mysql-slave  
 **labels:  
 name:** nacos-mysql-slave  
**spec:  
 replicas:** 1  
 **selector:  
 name:** nacos-mysql-slave  
 **template:  
 metadata:  
 labels:  
 name:** nacos-mysql-slave  
 **spec:  
 containers:** - **name:** slave  
 **image:** nacos/nacos-mysql-slave:latest  
 **ports:** - **containerPort:** 3306  
 **volumeMounts:** - **mountPath:** /var/lib/mysql  
 **name:** nacos-mysql-slave-data  
 **env:** - **name:** MYSQL\_ROOT\_PASSWORD  
 **valueFrom:  
 configMapKeyRef:  
 name:** nacos-mysql-cm  
 **key:** nacos.mysql.slave.root.password  
 - **name:** MYSQL\_REPLICATION\_USER  
 **valueFrom:  
 configMapKeyRef:  
 name:** nacos-mysql-cm  
 **key:** nacos.mysql.replication.user  
 - **name:** MYSQL\_REPLICATION\_PASSWORD  
 **valueFrom:  
 configMapKeyRef:  
 name:** nacos-mysql-cm  
 **key:** nacos.mysql.replication.password  
 **volumes:** - **name:** nacos-mysql-slave-data  
 **nfs:  
 path:** /nfs/nacos-mysql-slave  
 **server:** 192.168.84.48  
---  
**apiVersion:** v1  
**kind:** Service  
**metadata:  
 name:** mysql-slave**labels:  
 name:** mysql-slave  
**spec:  
 ports:** - **port:** 3306  
 **targetPort:** 3306  
 **selector:  
 name:** nacos-mysql-slave

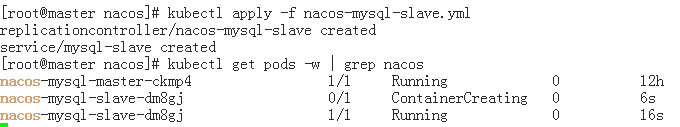
2、创建资源

kubectl apply -f nacos-mysql-slave.yml

查看创建过程和SVC

kubectl get pods -w | grep nacos

kubectl describe service/mysql-slave



## 四、部署Nacos服务端

1、创建Nacos Server资源文件

vim nacos-server.yml

内容如下

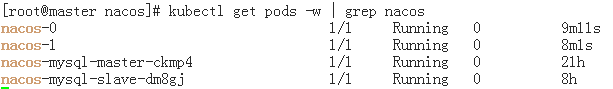
---  
**apiVersion:** v1  
**kind:** Service  
**metadata:  
 name:** nacos-headless  
 **labels:  
 app:** nacos  
 **annotations:  
 service.alpha.kubernetes.io/tolerate-unready-endpoints: "true"  
spec:  
 ports:** - **port:** 8848  
 **name:** server  
 **targetPort:** 8848  
 **clusterIP:** None  
 **selector:  
 app:** nacos  
---  
**apiVersion:** apps/v1  
**kind:** StatefulSet  
**metadata:  
 name:** nacos  
**spec:  
 serviceName:** nacos-headless  
 **replicas:** 2 *# 测试服务器资源有限，正式环境可改为3* **template:  
 metadata:  
 labels:  
 app:** nacos  
 **annotations:  
 pod.alpha.kubernetes.io/initialized: "true"  
 spec:  
 affinity:  
 podAntiAffinity:  
 requiredDuringSchedulingIgnoredDuringExecution:** - **labelSelector:  
 matchExpressions:** - **key: "app"  
 operator:** In  
 **values:** - nacos  
 **topologyKey: "kubernetes.io/hostname"  
 serviceAccountName:** nfs-client-provisioner  
 **initContainers:** - **name:** peer-finder-plugin-install  
 **image:** nacos/nacos-peer-finder-plugin:latest  
 **imagePullPolicy:** IfNotPresent *# 官方的为Always，作了修改* **volumeMounts:** - **mountPath: "/home/nacos/plugins/peer-finder"  
 name:** plugindir  
 **containers:** - **name:** nacos  
 **imagePullPolicy:** IfNotPresent  
 **image:** nacos/nacos-server:1.1.4  
 **resources:  
 requests:  
 memory: "2Gi"  
 cpu: "500m"  
 limits:  
 memory: "2Gi"  
 cpu: "1000m"  
 ports:** - **containerPort:** 8848  
 **name:** client-port  
 **env:** - **name:** NACOS\_REPLICAS  
 **value: "2"** *# 测试服务器资源有限，正式环境改为3，部署3个nacos server* - **name:** SERVICE\_NAME  
 **value: "nacos-headless"** - **name:** POD\_NAMESPACE  
 **valueFrom:  
 fieldRef:  
 apiVersion:** v1  
 **fieldPath:** metadata.namespace  
 - **name:** MYSQL\_MASTER\_SERVICE\_DB\_NAME  
 **valueFrom:  
 configMapKeyRef:  
 name:** nacos-mysql-cm  
 **key:** nacos.mysql.master.db  
 - **name:** MYSQL\_MASTER\_SERVICE\_PORT  
 **valueFrom:  
 configMapKeyRef:  
 name:** nacos-mysql-cm  
 **key:** nacos.mysql.master.port  
 - **name:** MYSQL\_SLAVE\_SERVICE\_PORT  
 **valueFrom:  
 configMapKeyRef:  
 name:** nacos-mysql-cm  
 **key:** nacos.mysql.slave.port  
 - **name:** MYSQL\_MASTER\_SERVICE\_USER  
 **valueFrom:  
 configMapKeyRef:  
 name:** nacos-mysql-cm  
 **key:** nacos.mysql.master.user  
 - **name:** MYSQL\_MASTER\_SERVICE\_PASSWORD  
 **valueFrom:  
 configMapKeyRef:  
 name:** nacos-mysql-cm  
 **key:** nacos.mysql.master.password  
 - **name:** NACOS\_SERVER\_PORT  
 **value: "8848"** - **name:** PREFER\_HOST\_MODE  
 **value: "hostname"  
 readinessProbe:  
 httpGet:  
 port:** client-port  
 **path:** /nacos/v1/console/health/readiness  
 **initialDelaySeconds:** 60  
 **timeoutSeconds:** 3  
 **livenessProbe:  
 httpGet:  
 port:** client-port  
 **path:** /nacos/v1/console/health/liveness  
 **initialDelaySeconds:** 60  
 **timeoutSeconds:** 3  
 **volumeMounts:** - **name:** plugindir  
 **mountPath:** /home/nacos/plugins/peer-finder  
 - **name:** datadir  
 **mountPath:** /home/nacos/data  
 - **name:** logdir  
 **mountPath:** /home/nacos/logs**volumeClaimTemplates:** - **metadata:  
 name:** plugindir  
 **spec:  
 storageClassName:** managed-nfs-storage  
 **accessModes:** [ **"ReadWriteMany"** ]  
 **resources:  
 requests:  
 storage:** 5Gi  
 - **metadata:  
 name:** datadir  
 **spec:  
 storageClassName:** managed-nfs-storage  
 **accessModes:** [ **"ReadWriteMany"** ]  
 **resources:  
 requests:  
 storage:** 5Gi  
 - **metadata:  
 name:** logdir  
 **spec:  
 storageClassName:** managed-nfs-storage  
 **accessModes:** [ **"ReadWriteMany"** ]  
 **resources:  
 requests:  
 storage:** 5Gi  
 **selector:  
 matchLabels:  
 app:** nacos  
---**apiVersion:** v1  
**kind:** Service  
**metadata:  
 name:** nacos-service  
**spec:  
 ports:** - **port:** 8001 *# 对集群内部端口为8001* **targetPort:** 8848  
 **nodePort:** 30000 *# 对外端口为30000* **type:** NodePort *# 采用NodePort模式暴露服务* **selector:  
 app:** nacos

PS：由于测试服务器资源有限，只有两个node节点，nacos server的pod互斥调度需要将三个实例分别部署在三个node节点，所以这里将实例数量改为了2，正式环境可根据需要修改成3，资源限制如内存等可调大些。

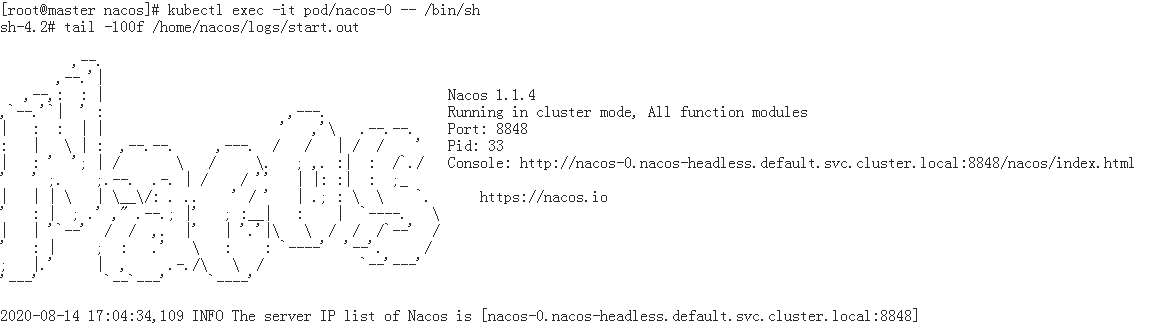
2、创建资源

kubectl apply -f nacos-server.yml

kubectl get pods -w | grep nacos



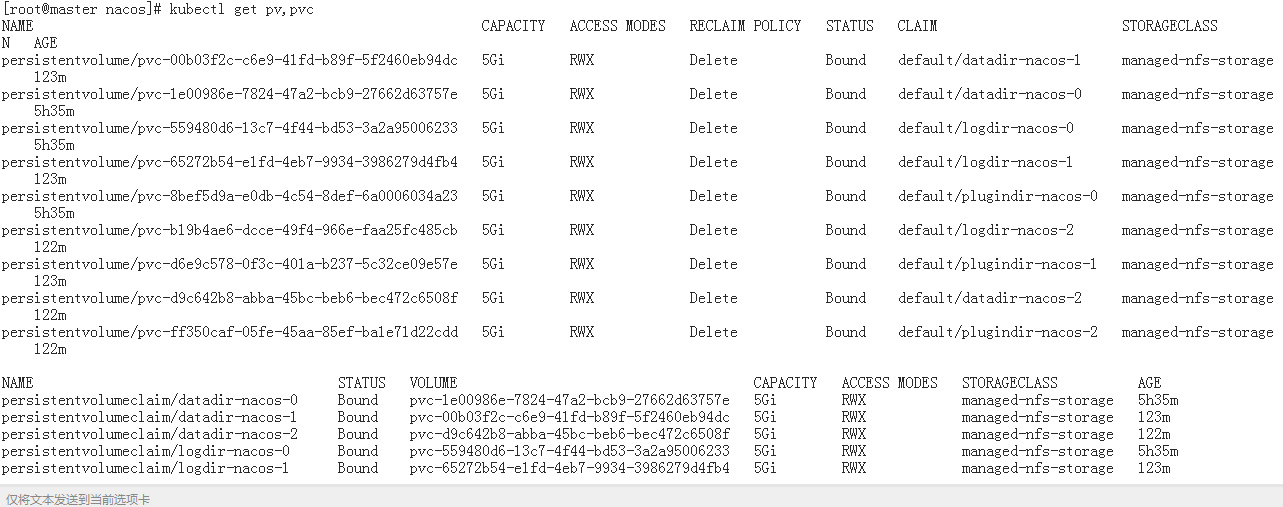
3、查看资源启动情况



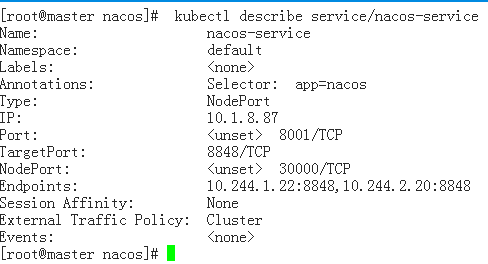


正常启动并处于集群模式

kubectl get pv,pvc查看PV和PVC的STATUS都为Bound状态



kubectl describe service/nacos-service

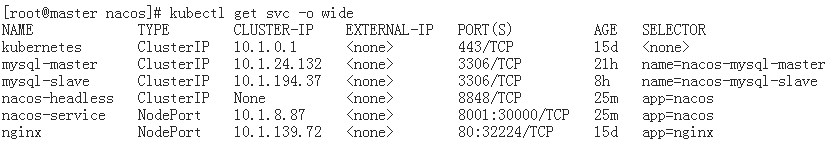


已成功绑定两个Endpoints

浏览器访问 <http://192.168.84.49:30000/nacos>用户名和密码都为ConfigMap中设置的nacos

## 五、kubernetes搭建nacos集群的BUG修改

通过nacos-sevice来测试下Nacos集群心跳服务



nacos-service对应的IP为：10.1.8.87，往这个地址发送心跳

curl 10.1.8.87:8001/nacos/v1/ns/instance/beat

返回结果：



官方BUG，地址为：<https://github.com/nacos-group/nacos-k8s/issues/54>

解决方案是手动把nacos-0删除

kubectl delete pods nacos-0

等待kubernetes恢复nacos-0实例，再次发送心跳，不会再有信息返回（说明心跳发送成功）



登录nacos查看节点列表



Nacos集群搭建完成