Day10 kubernetes笔记

# 1、课程回顾

1）configmap 环境变量，配置文件存储（配置中心）

2）secret 加密数据

3）volumes 数据卷

本地卷

网络卷：nfs

volumeMounts:

- name: wwwroot

mountPath: /usr/share/nginx/html

ports:

- containerPort: 80

volumes:

- name: wwwroot

nfs:

server: 192.168.66.13

path: /opt/k8s/wwwroot

今日课程主题：有状态服务部署

1、Pv [持久化数据卷]

2、Pvc

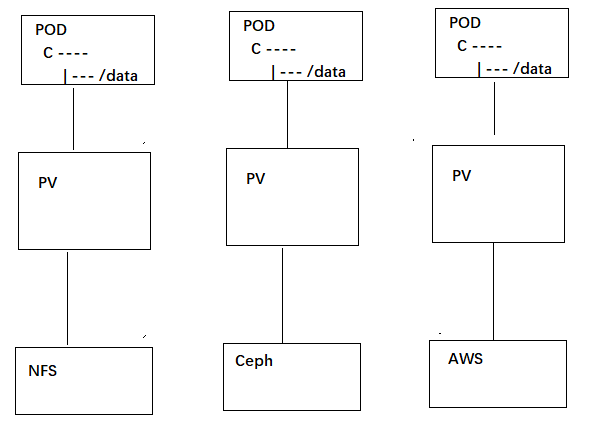
3、pod \* pv \* pvc

4、headless service

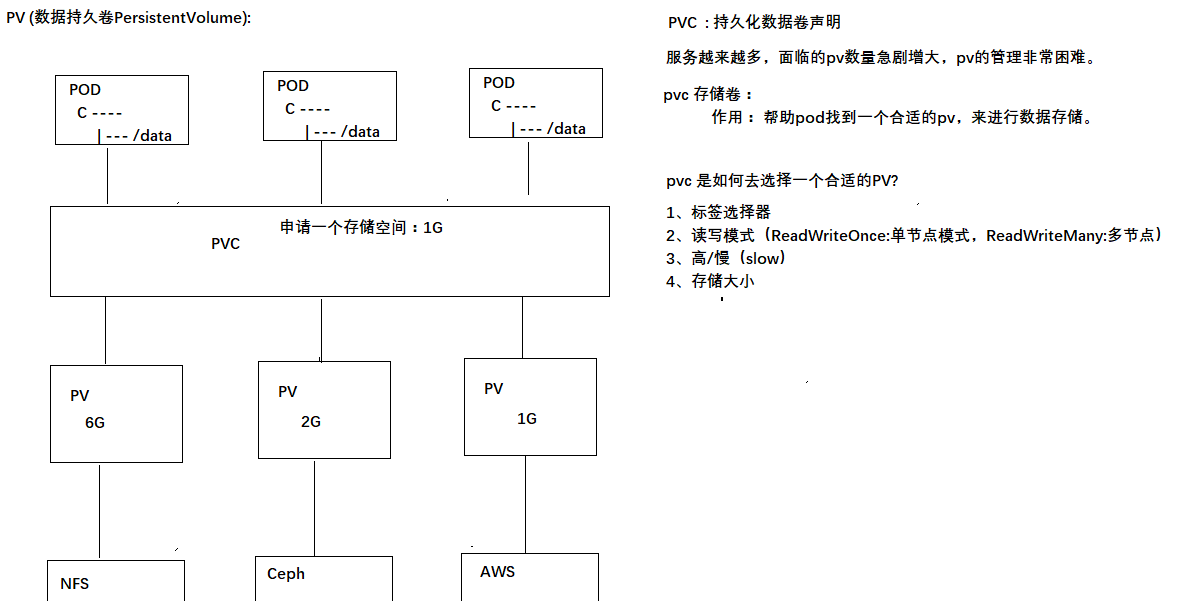
5、statefulset \* pv \* pvc

# 2、Pv

Pv [持久化数据卷]： 用来屏蔽pod对底层物理存储介质的关联性。Pod不用直接和底层物理存储介质进行交互，不需要关心底层到底是如何存储的，pod只需要关联pv进行数据的挂载即可。



Pv pvc关系：



# 3、实践操作

1、构建网络存储介质（nfs文件系统），nfs处于启动状态

2、先创建pv持久化数据卷(k8s资源对象)，使用nfs网络存储设备挂载数据

# 先构建第一个pv

apiVersion: v1

kind: PersistentVolume

metadata:

name: my-pv1

spec:

capacity:

storage: 5Gi

accessModes:

- ReadWriteMany

nfs:

path: /opt/k8s/demo1

server: 192.168.66.13

# 第二个pv

apiVersion: v1

kind: PersistentVolume

metadata:

name: my-pv2

spec:

capacity:

storage: 10Gi

accessModes:

- ReadWriteMany

nfs:

path: /opt/k8s/demo2

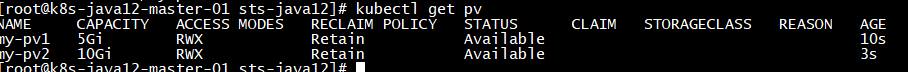
server: 192.168.66.13

执行创建持久化数据卷：

Kubectl apply -f my-pv1.yaml

Kubectl apply -f my-pv2.yaml

产生了2个持久化数据卷资源对象，说明了pv就是把持久化数据存储方式（物理设备）抽象成了k8s的资源对象。



注意：pv持久化的数据卷通常都是由运维来进行维护的。开发更多是服务部署pod,pvc这块维护部署。

# 4、Pvc

创建pod，同时通过pvc选择一个合适的pv存储数据。

apiVersion: v1

kind: Pod

metadata:

name: my-pod

spec:

containers:

- name: nginx

image: hub.kaikeba.com/java12/myapp:v1

ports:

- containerPort: 80

volumeMounts:

- name: www

mountPath: /usr/share/nginx/html # 把volumes挂载的数据挂载此目录下

volumes:

- name: www

persistentVolumeClaim:

claimName: my-pvc # 通过此名称关联相应的pvc

---

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: my-pvc

spec: # spec所描述的内容： 表示向pv申请一个合适的存储介质

accessModes:

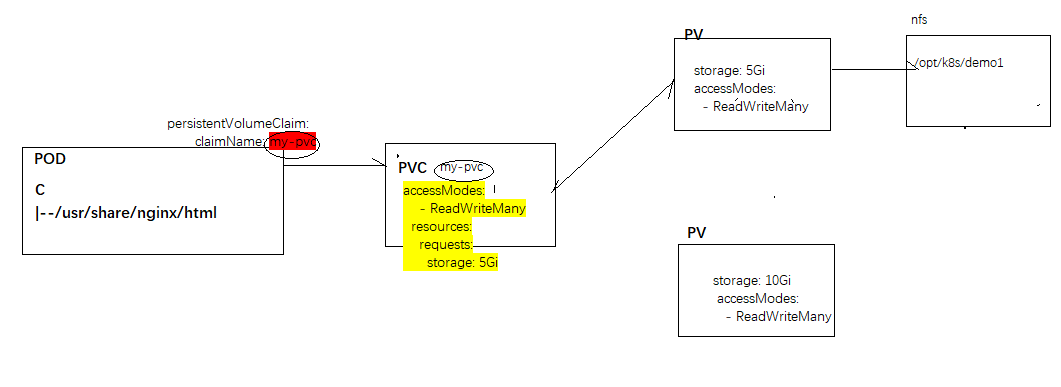
- ReadWriteMany

resources:

requests:

storage: 5Gi

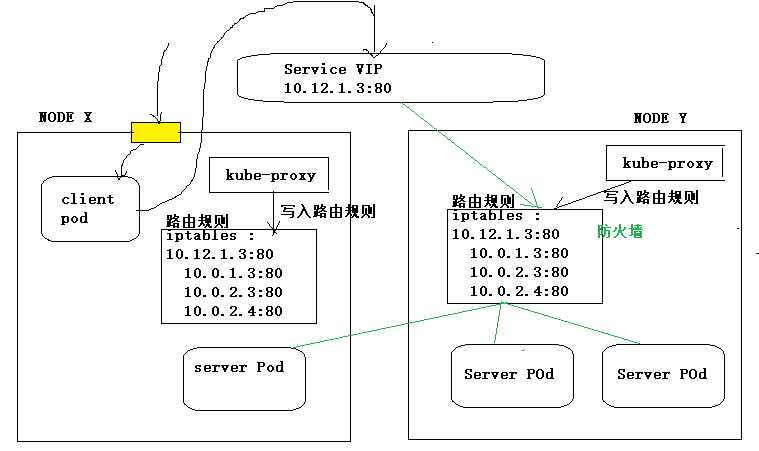
pod ，pvc ，pv ,nfs 关联关系：



# 5、Headless service

## 5.1、Services

之前delployment有状态服务部署service，有局域网的ip地址（VIP），可以在局域网进行通信，对外部的访问请求进行转发，通过iptables,ipvs路由规则把请求转发给相应的pod.

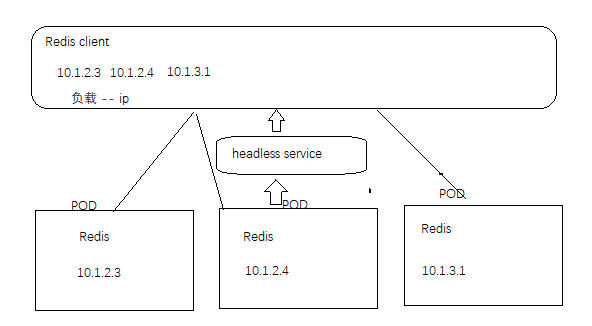


## 5.2、headless

无头service，此service服务没有ip(VIP)地址，是一种特殊的service，是用来对有状态服务的部署提供的Sevice.

例如：

Redis,kafka等等软件，客户端需要的是一组pod对应的ip地址。



Headless service 无头服务，把一组相关的pod的ip地址暴露给客户端，相关访问pod服务的负载由客户端来实现。

Headless service 暴露了pod ip地址，使得在局域网可以直接访问pod,实现了pod与pod之间的互相访问。

# 6、Statefulset

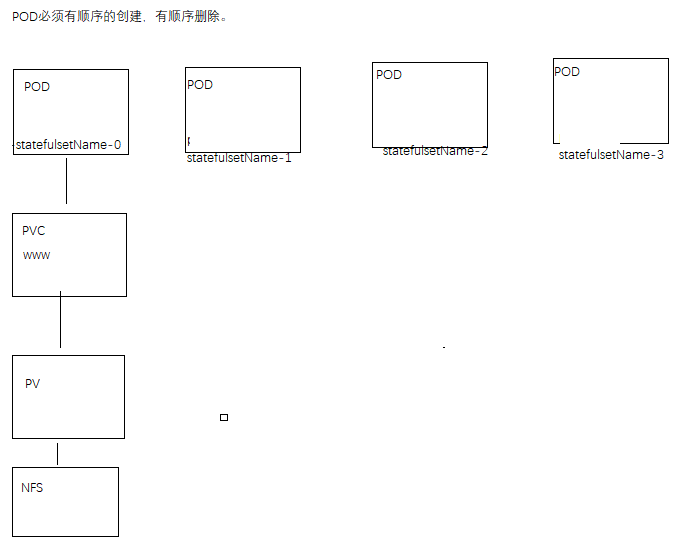
StatefullSet 是为了解决有状态服务的问题（对应Deployments 和 ReplicaSets 是为无状态服务而设计），其应用场景包括：

（1） 稳定的持久化存储，即Pod重新调度后还是能访问的相同持久化数据，基于PVC来实现

（2）稳定的网络标志，及Pod重新调度后其 PodName 和 HostName 不变，基于Headlesss Service（即没有 Cluster IP 的 Service）来实现。

（3）有序部署，有序扩展，即Pod是有顺序的，在部署或者扩展的时候要依据定义的顺序依次进行（即从 0 到 N-1,在下一个Pod运行之前所有之前的Pod必须都是Running 和 Ready 状态），基于 init containers 来实现。

（4）有序收缩，有序删除（即从N-1 到 0）



Satefulset部署：

声明方式部署：（pvc自定义pv,pod的name名称： statefulsetName-0…N）

apiVersion: v1

kind: Service

metadata:

name: nginx

labels:

app: nginx

spec:

ports:

- port: 80

name: web

clusterIP: None

selector:

app: nginx

---

apiVersion: apps/v1

kind: StatefulSet

metadata:

name: web

spec:

selector:

matchLabels:

app: nginx

serviceName: "nginx"

replicas: 3

template:

metadata:

labels:

app: nginx

spec:

terminationGracePeriodSeconds: 10

containers:

- name: nginx

image: hub.kaikeba.com/java12/myapp:v1

ports:

- containerPort: 80

name: web

volumeMounts:

- name: www

mountPath: /usr/share/nginx/html

volumes:

- name: www

persistentVolumeClaim:

claimName: my-pvc

---

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: my-pvc

spec:

accessModes:

- ReadWriteMany

resources:

requests:

storage: 1Gi

模板部署：volumeClaimTemplates （帮助我们自动创建pvc,不需要我们手动定义pvc）

apiVersion: v1

kind: Service

metadata:

name: nginx

labels:

app: nginx

spec:

ports:

- port: 80

name: web

apiVersion: v1

kind: Service

metadata:

name: nginx

labels:

app: nginx

spec:

ports:

- port: 80

name: web

storageClassName: "my-storage-class"

clusterIP: None

selector:

app: nginx

---

apiVersion: apps/v1

kind: StatefulSet

metadata:

name: web

spec:

selector:

matchLabels:

app: nginx

serviceName: nginx

replicas: 3

template:

metadata:

labels:

app: nginx

spec:

terminationGracePeriodSeconds: 10

containers:

- name: nginx

image: hub.kaikeba.com/java12/myapp:v1

ports:

- containerPort: 80

name: web

volumeMounts:

- name: www

mountPath: /usr/share/nginx/html

volumeClaimTemplates:

- metadata:

name: www

spec:

accessModes: [ "ReadWriteOnce" ]

resources:

requests:

storage: 1Gi

volumeClaimTemplates方式部署，自动创建pvc,然后绑定相应的pv持久化的数据卷。

特点： 默认创建与pod相对应的数量的pvc,一个pod对应一个pvc,然后挂载一个pv

# 7、storageclass

Pvc的自动化实现方式：volumeClaimTemplates

Pv自动化： storageClass实现，可以根据pvc需求，自动构建相对应的pv持久化存储卷，进一步简化运维管理成本。

参考note资料。