Kubernetes存储和负载均衡的实现

万绍远@Rancher Labs

RANCHER

2018.05

应用类型

- 无状态应用
- 有状态应用
- 有状态集群应用

volume类型

静态供给volume

- emptyDir
- Hostpath
- storage-provide
- Persistent volume

动态供给volume

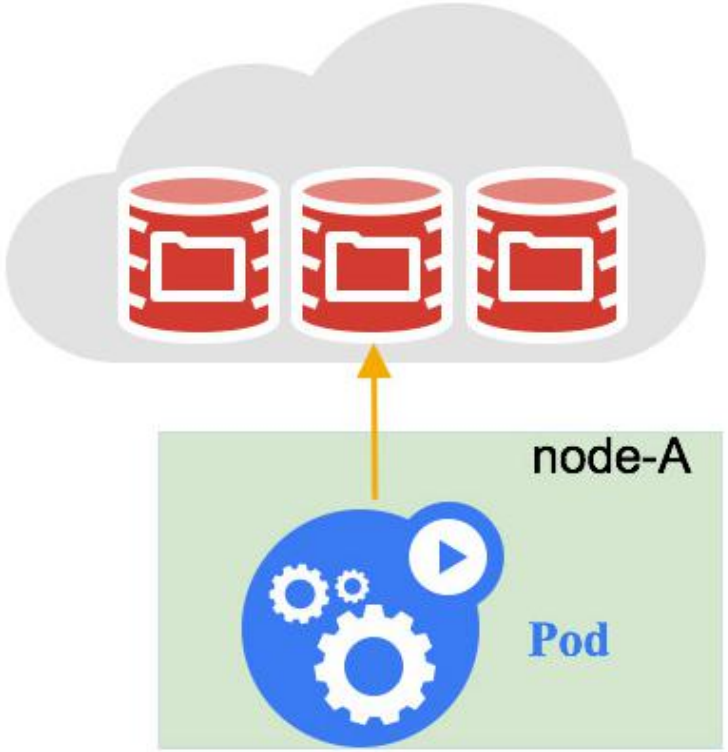
- Storage-class

静态供给volume

- emptyDir

emptydir：临时空目录，与pod紧密联接在创建pod是创建，在删除这个pod时，也会自动删除，pod迁移到其他节点数据会丢失。

用途：pod内多个container同享一个目录



静态供给volume

- Hostpath

根docker bind-mount与宿主机目录1:1映射，这类存储卷，当数据迁移到其他节点后，就会造成数据丢失。
用途：根DaemonSet配合使用如EFK，中fluentd 根容器日志目录映射，来达到收集日志效果。

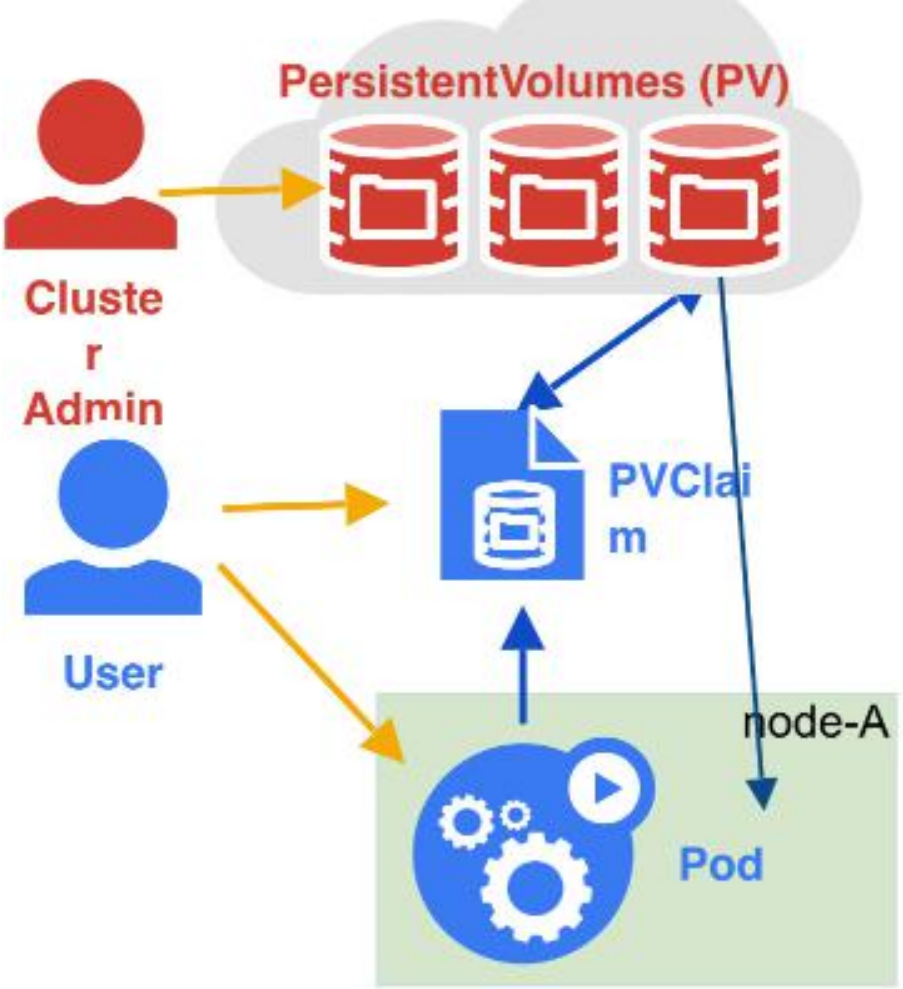
- Storage-private

不使用host存储空间，使用公有云厂商对象存储或分布式存储提供volume

静态供给volume

- Persistent volume

volume虽然能提供很好的数据持久化，但在可管理性上，还是不足的，要使用volume，用户必须，知道当前的volume信息和提前创建好对应的volume，kubernetes推荐使用pv、pvc来解决存储持久化的问题。因此kubernetes给出的解决方案是pv(persistent volume)、pvc(persistent volume Claim)



静态供给volume

- Persistent volume

pv的回收模式

persistentVolumeReclaimPolicy 为当pvc删除后pv的回收策略
Retain – pvc删除后pv和数据仍然保留但此时不可以在创建pvc了，需要管理员手工回收。
Recycle – pvc删除后回自动起一个pod将pv内的数据全部清空，可以创建新的pvc。
Delete – 删除 Storage Provider 上的对应存储资源，例如 AWS EBS、GCE PD、Azure Disk、OpenStack Cinder Volume 等。

静态供给volume

- Persistent volume

pv的访问模式

在pvc绑定pv时通常根据两个条件绑定，一个是存储的大小，另外一个是存储的模式

ReadWriteOnce(RWO)：可读写模式支持单节点挂载

ReadOnlyMany(ROX)： 只读模式支持多节点挂载

ReadWriteMany(RWX)：可读写模式支持多节点挂载，目前只有少数存储支持这种方式，像ceph-rbd目前只能当个节点挂载

静态供给volume

- Persistent volume

Kubernetes支持的pv类型

Volume Plugin	ReadWriteOnce	ReadOnlyMany	ReadWriteMany
AWSElasticBlockStore	✓	-	-
AzureFile	✓	✓	✓
AzureDisk	✓	-	-
CephFS	✓	✓	✓
Cinder	✓	-	-
FC	✓	✓	-
FlexVolume	✓	✓	-
Flocker	✓	-	-
GCEPersistentDisk	✓	✓	-
Glusterfs	✓	✓	✓
HostPath	✓	-	-
iSCSI	✓	✓	-
PhotonPersistentDisk	✓	-	-
Quobyte	✓	✓	✓
NFS	✓	✓	✓
RBD	✓	✓	-
VsphereVolume	✓	-	- (works when pods are collocated)
PortworxVolume	✓	-	✓
ScaleIO	✓	✓	-
StorageOS	✓	-	-

动态供给volume

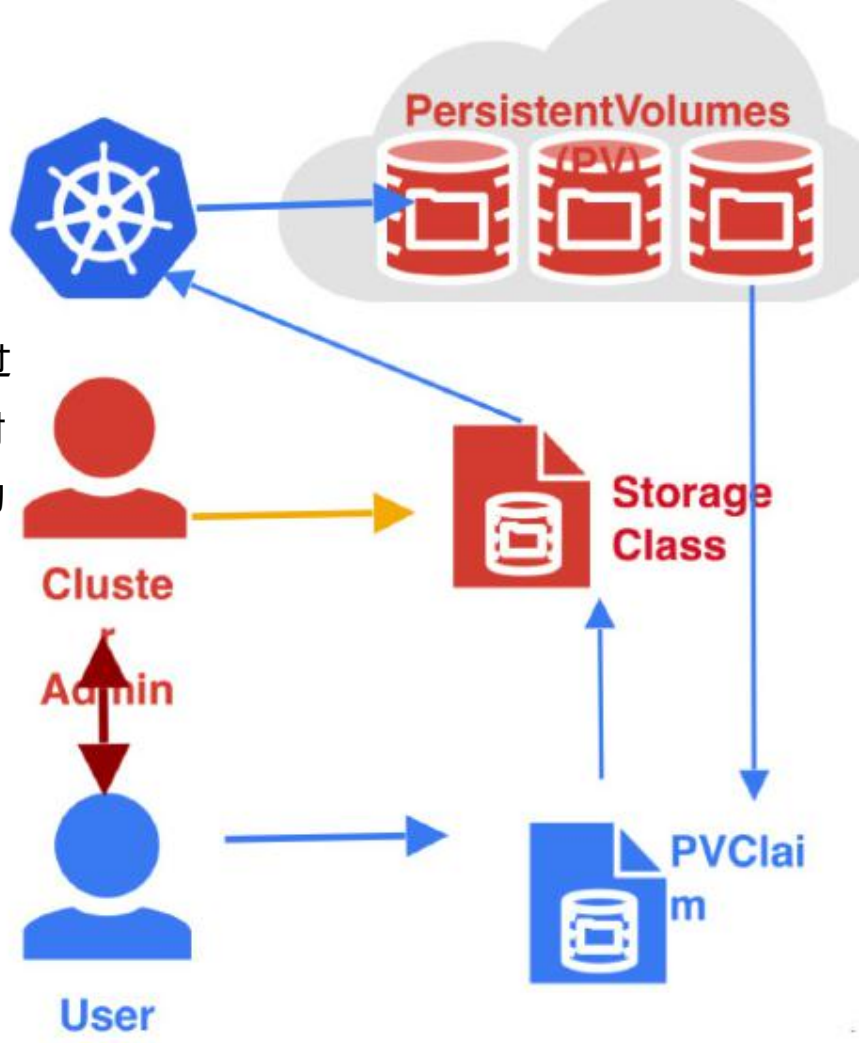
- Storage-class

直接使用pv方式都是静态供给，需要管理员提前将pv创建好，然后再与pvc绑定，在kubernetes中动态卷是通过storage-class去实现的。配好storage-class与backend对接，当没有满足pvc条件的pv时，storage-class会动态的去创建一个pv

动态卷的优势

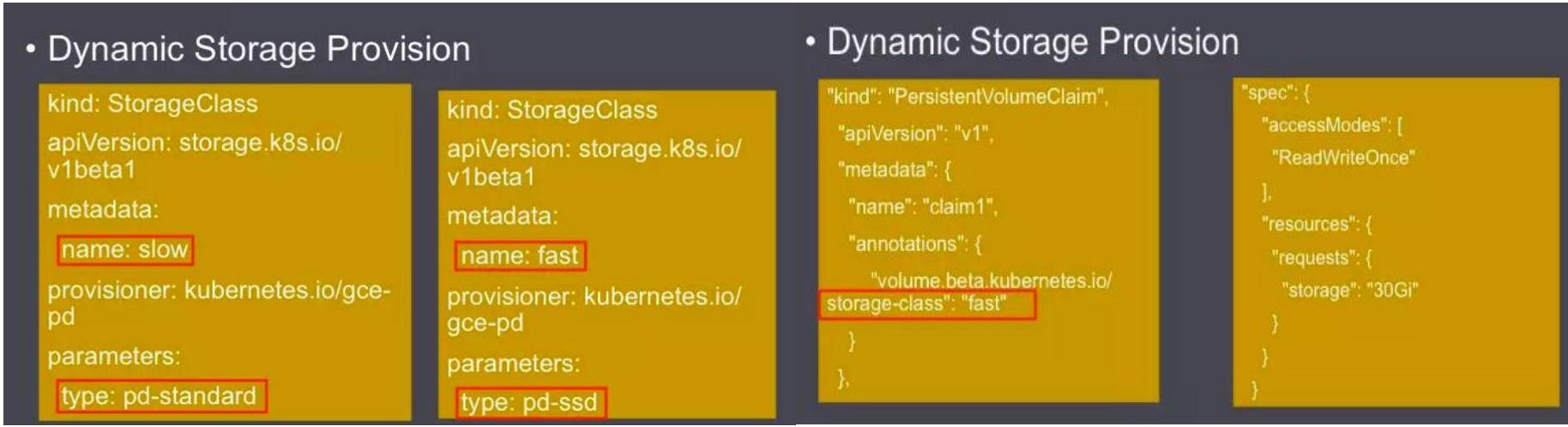
1、不需要提前创建好pv，提高效率和资源利用率

2、封装不同的存储类型给pvc使用，在StorageClass出现以前，PVC绑定一个PV只能根据两个条件，一个是存储的大小，另一个是访问模式。在StorageClass出现后，等于增加了一个绑定维度。



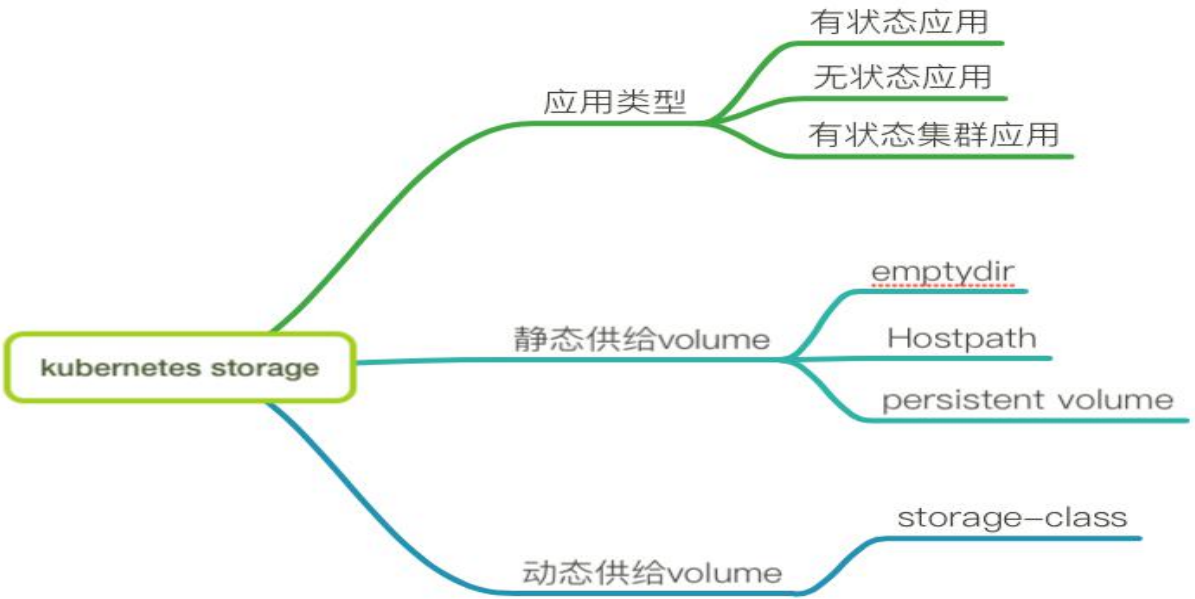
动态供给volume

- Storage-class



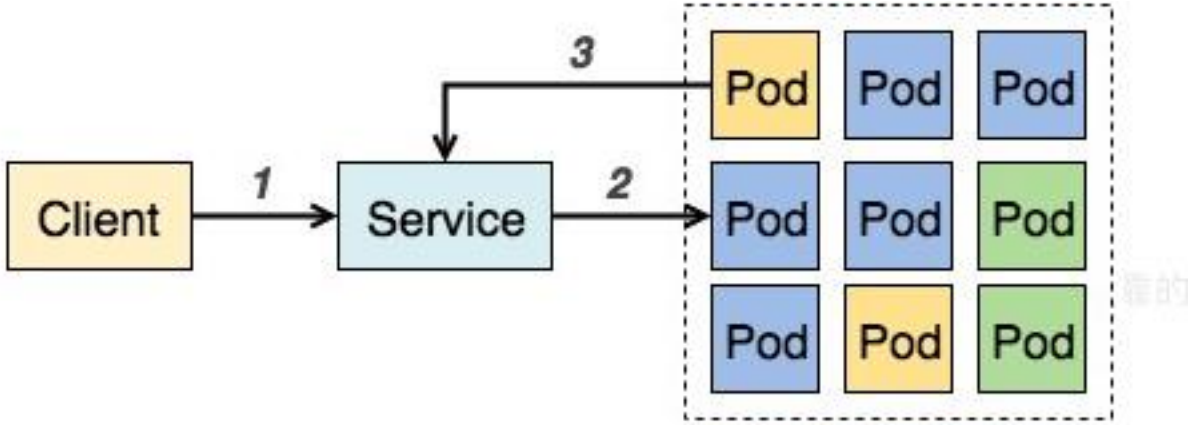
RANCHER

总 结


kubernetes storage
应用类型
有状态应用
无状态应用
有状态集群应用
静态供给volume
emptydir
Hostpath
persistent volume
动态供给volume
storage-class

Kuberne负载均衡的实现和使用

pod是随时变化的，动态的伸缩，所以应用程序直接连接pod是不可靠的，所以kubernetes引入service，service是为一组功能相同的pod提供统一入口。



RANCHER

Kuberne负载均衡的实现和使用

端口类型

port

Service暴露出来的内部访问端口

targetPort

pod暴露出来的端口

NodePort

NodePort是kubernetes提供外部访问service的暴露端口，它暴露在集群所有宿主机上，默认端口范围为30000~32767

```
kind: Service
apiVersion: v1
metadata:
  name: httpd-deployment
spec:
  type: NodePort
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30080
  selector:
    app: http
```


RANCHER

Kuberne负载均衡的实现和使用

服务类型

ClusterIP（默认)

使用clusterip类型的Service默认会使用一个集群内部ip，通过kube-proxy调用iptables创建规则，将流量转发到pod中，需要注意的是clusterip是一个virtual_ip没有真正的网络设备绑定，所以是ping不通它的，直接在集群内部的访问就好。

NodePort

使用NodePort类型的Service时会在集群内部所有host上暴露一个端口用于外部访问默认端口范围30000~32767

LoadBlancer

使用loadblance类型时，会向cloud provider申请映射到service本身的负载均衡，比如AWS、google cloud、azure。

Kuberne负载均衡的实现和使用

外部访问pod

NodePort

loadblance

ingress

Service的扩展

ingress就是可以暴露内部访问工作在7层的负载均衡器，根据域名或服务名对后端Service进行端口转发，并且能根据后端Service的变化，动态刷新配置。

为什么使用ingress？

因为如果Service使用nodeport暴露host的端口方式去访问应用的话，当Service有多个时，会造成端口管理上的混乱.

ingress组成？

Default-backend：用来将未知请求全部负载到这个默认后端上，这个默认后端会返回 404 页面。同时对ingress-controller进行健康状态检查。

ingress-controller ： 根kube-apiserver交互读取Service规则，生成反向代理规则

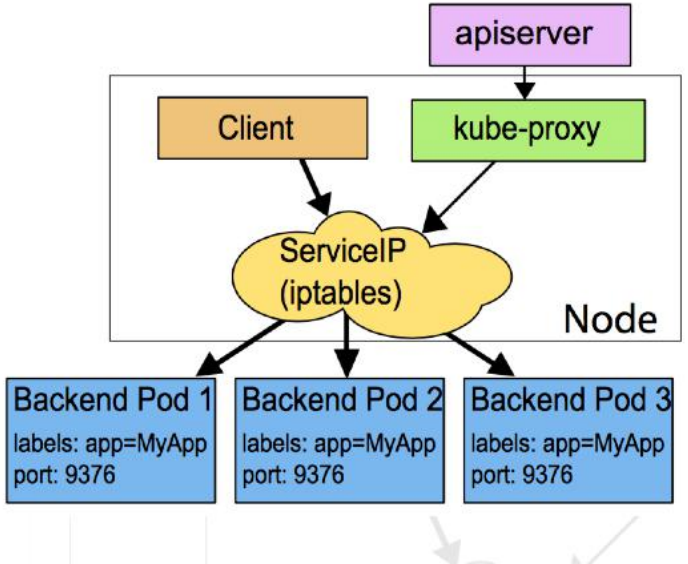
RANCHER

Kuberne负载均衡的实现和使用

kube-proxy 会检测 API Server 上对于 service 和 endpoint 的新增或者移除。对于每个新的 service，在每个 node 上，kube-proxy 都会设置相应的 iptables 的规则来记录应该转发的地址。当一个 service 被删除的时候，kube-proxy 会在所有的 pod 上移除这些 iptables 的规则。默认转发规则是round robin 或session affinity

演示

kube-proxy 模式：userspace、iptables、ipvs(beta)



RANCHER

Kuberne负载均衡的实现和使用

服务发现

两种方式

环境变量

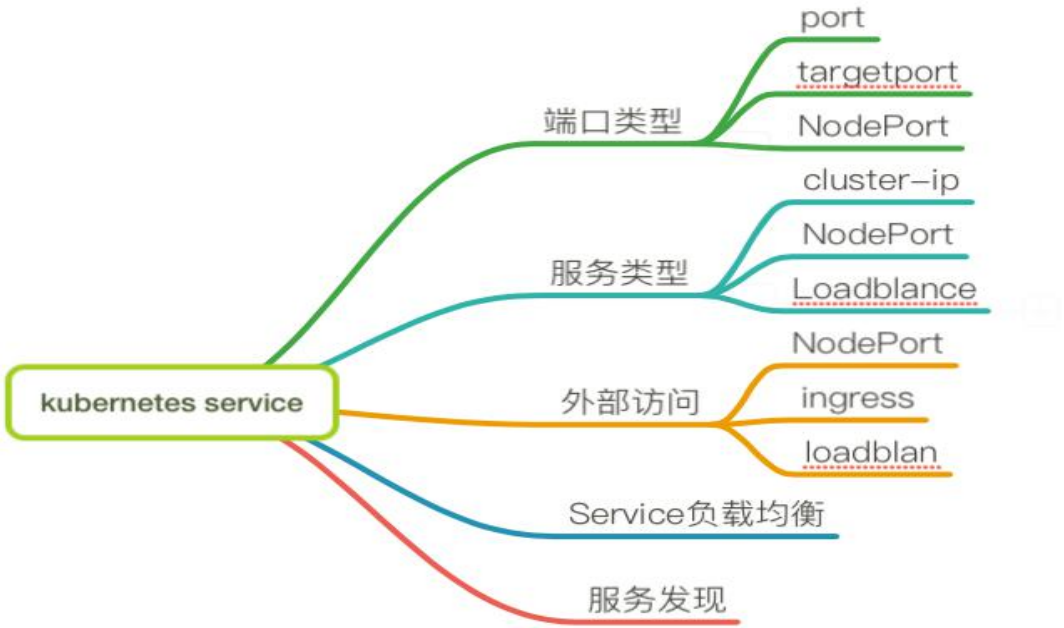
DNS:1.10版之前是通过kube-dns实现服务发现，1.10版后可以用CoreDns替代kube-dns做服务发现

创建一个Service后，kubernetes会自动将servername做为服务名，添加一条dns记录

比如上面那个例子Service名为httpd-deployment,那么其他pod要访问直接访问httpd-deployment就会解析到对应cluster-ip,跨namespace的只需要在域名后接上.namespace_name 如httpd-deployment.default

Kuberne负载均衡的实现和使用

总　结



RANCHER

Thanks