

Kubernetes On Alibaba Cloud

谢瑶瑶 @ 阿里云

2017-10-09

阿里云容器服务介绍

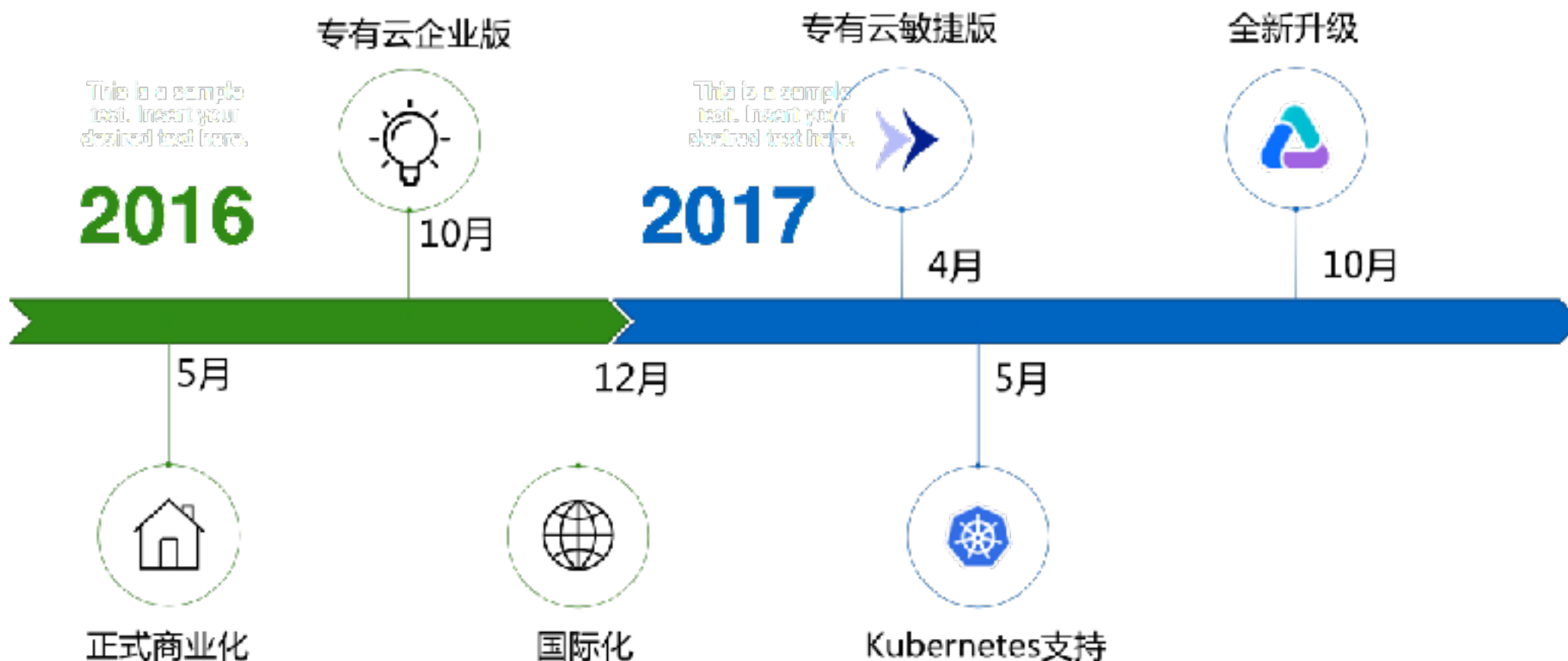
Kubernetes集群

阿里云CloudProvider

主办：



阿里云容器服务发展



阿里云容器服务

生命周期管理

集群与应用：创建、删除，自动扩缩容，故障恢复，健康监控

服务发现

提供四层、七层服务路由，支持阿里云SLB提供负载均衡

持久化存储

支持多种存储选择，阿里云盘，NAS网络文件存储

日志管理

支持对接阿里云日志系统

编排与调度

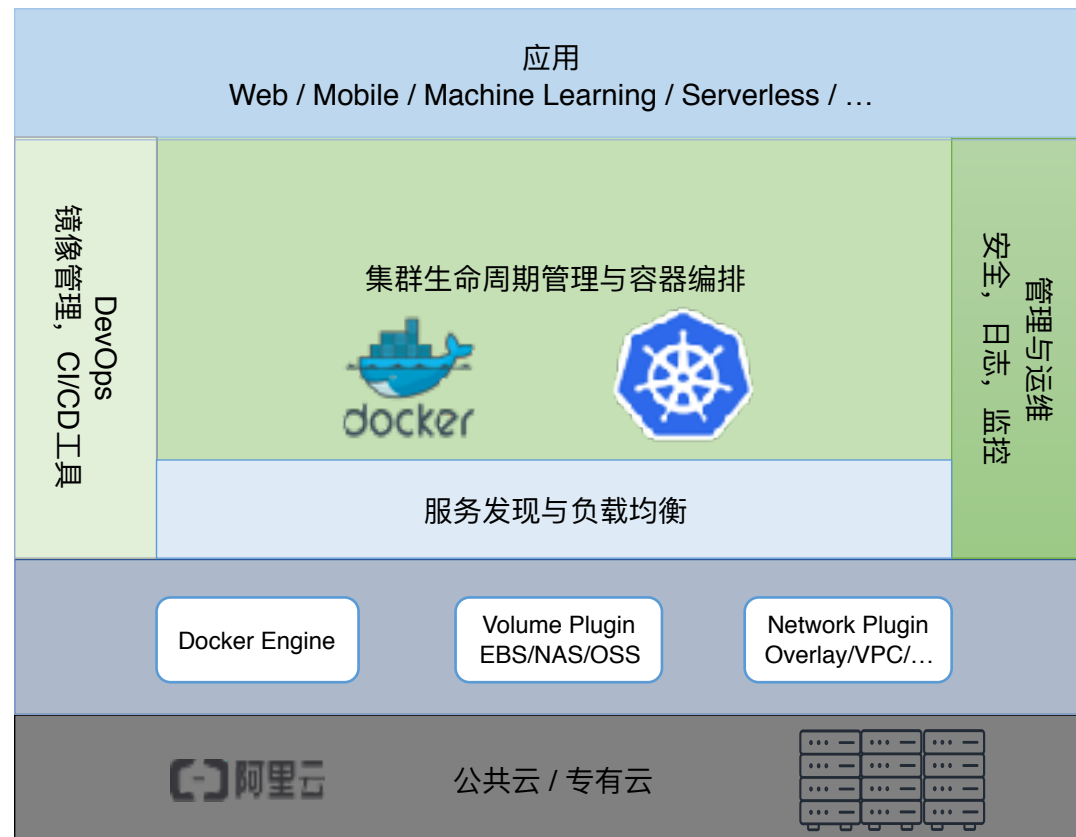
支持docker与kubernetes双核心编排引擎

网络管理

多种网络选择，VPC，vxlan

监控管理

集群监控，应用监控



主办:



容器服务Kubernetes与开源社区



Kubernetes
Out-of-tree CloudProvider



VPC网络
Flannel vpc driver



阿里云盘
Flexvolume driver



OSS存储
Registry 后端存储



日志
Fluentd-pilot日志

阿里云容器服务介绍

Kubernetes集群

阿里云CloudProvider

主办：



高可用的kubernetes用户集群

高可用

阿里云SLB为多副本组件提供统一负载均衡。
master的本地组件访问apiserver直接走本机IP，不经过内网SLB。

安全

ETCD节点之间通过证书认证，客户端通过证书访问etcd集群。

Kubernetes各个组件通过证书认证。

VPC网络提供安全隔离，自定义最小安全组策略控制节点访问。

云资源

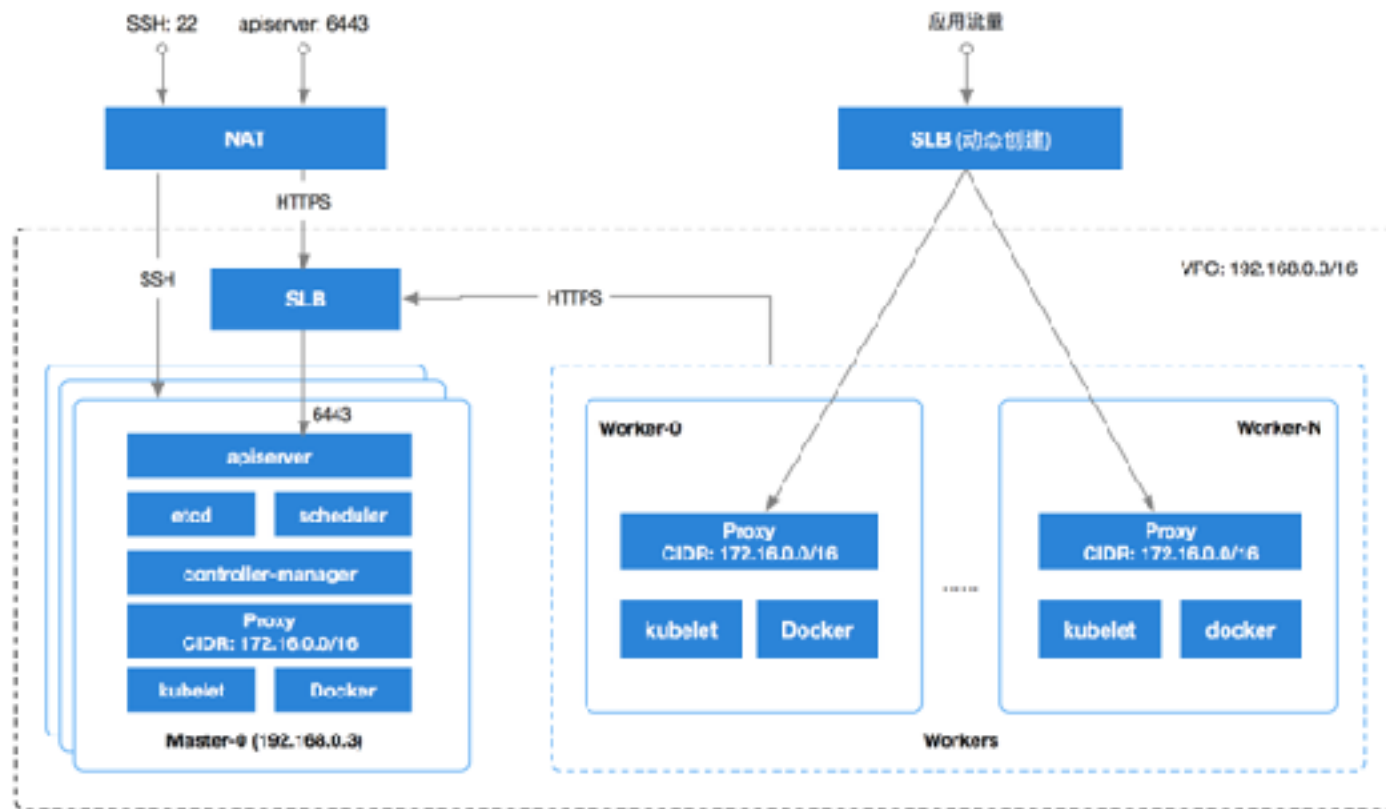
提供阿里云CloudProvider集成，轻松访问阿里云资源。

扩缩容

应用和集群动态扩缩容。

镜像加速

默认提供dockerhub官方镜像加速，解决官方镜像下载缓慢的问题。



一些建议

Kubeadm 支持部署单机开发测试集群

Cloud-Config路径问题

新版的Kubeadm为了安全原因，不在整体挂载/etc/kubernetes目录到组件容器中，同时也漏掉了cloud-config文件，造成升级故障。

镜来源问题

修改kubeadm可以使用非gcr.io来源的镜像。

处理证书方面

Cluster CA。

EXTRA CN 允许通过LoadBalancer访问。

处理kubeproxy问题

kubeadm生成的ds默认适合单集群，kubeproxy在访问apiserver时需要修改成LoadBalancer的地址。

kubeproxy在处理externalTraffic的时候，依赖hostname来判断Pod是否属于本地节点。

apiserver_count

kubernetes service 的endpoint会在多个master IP里面随机跳来跳去。

Dashboard权限

阿里云容器服务介绍

Kubernetes集群

阿里云CloudProvider

主办：



Kubernetes out-of-tree alibaba cloudprovider

CloudProvider

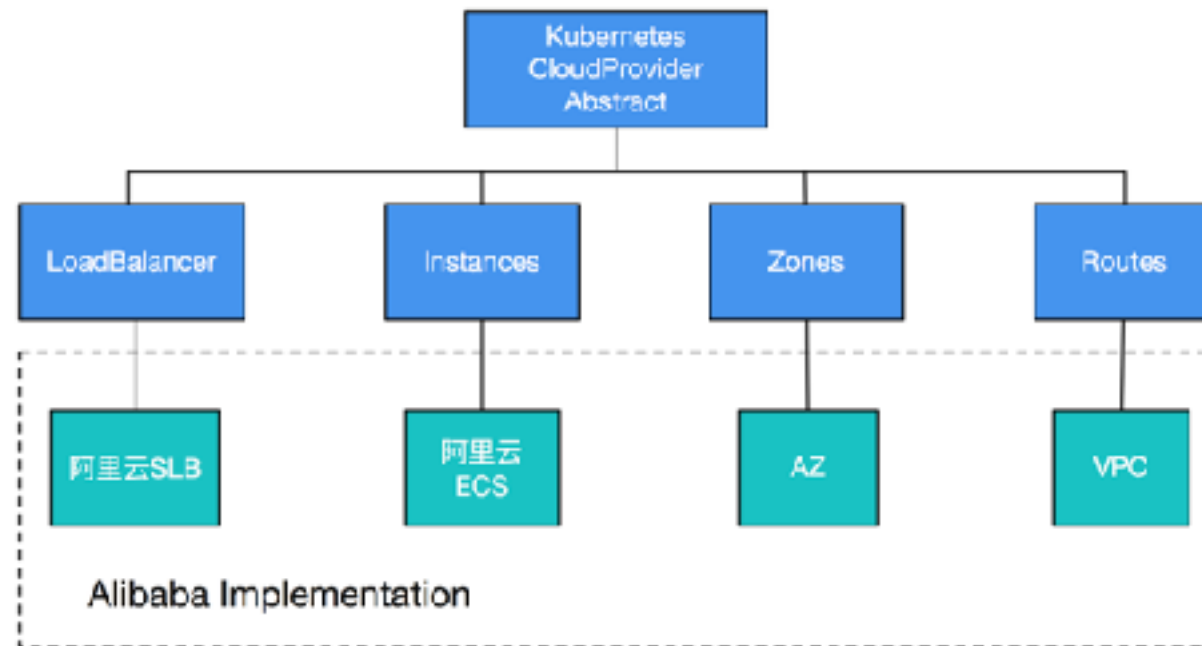
定义了一组接口，不同的云厂商通过实现这组接口来将本厂提供的各种服务集成到kubernetes集群中。

能力集成

网络路由配置、负载均衡动态配置、可用区管理、虚拟机管理

进化

In-tree-cloudprovider
cloud-out-of-tree cloud controller



进化In-Tree vs Out tree

传统 In-tree 问题

所有的云厂商的provider代码放在core kubernetes仓库可维护性低。
Cloud provider实现发布周期与kuberentes core耦合，无法快速迭代

解决方案

out-of-tree cloud providers

使用kubernetes的controller的概念来设计重构cloud provider, 即cloud-controller-manager。
将与云厂商实现相关代码移动到一个单独仓库，作为单独的二进制文件发布

路线图

Alpha: 1.7 => Beta: 1.8 => Stable: 1.10

开源

<https://github.com/AliyunContainerService/alibabacloud-controller-manager>

Kubernetes集群网络支持高效互联和混合云

#571 Add alibaba cloud VPC network support

CNI规范

定义了如何处理容器内网络设置。创建网卡分配IP，设置路由，DNS。Meta/Main

集群容器间网络

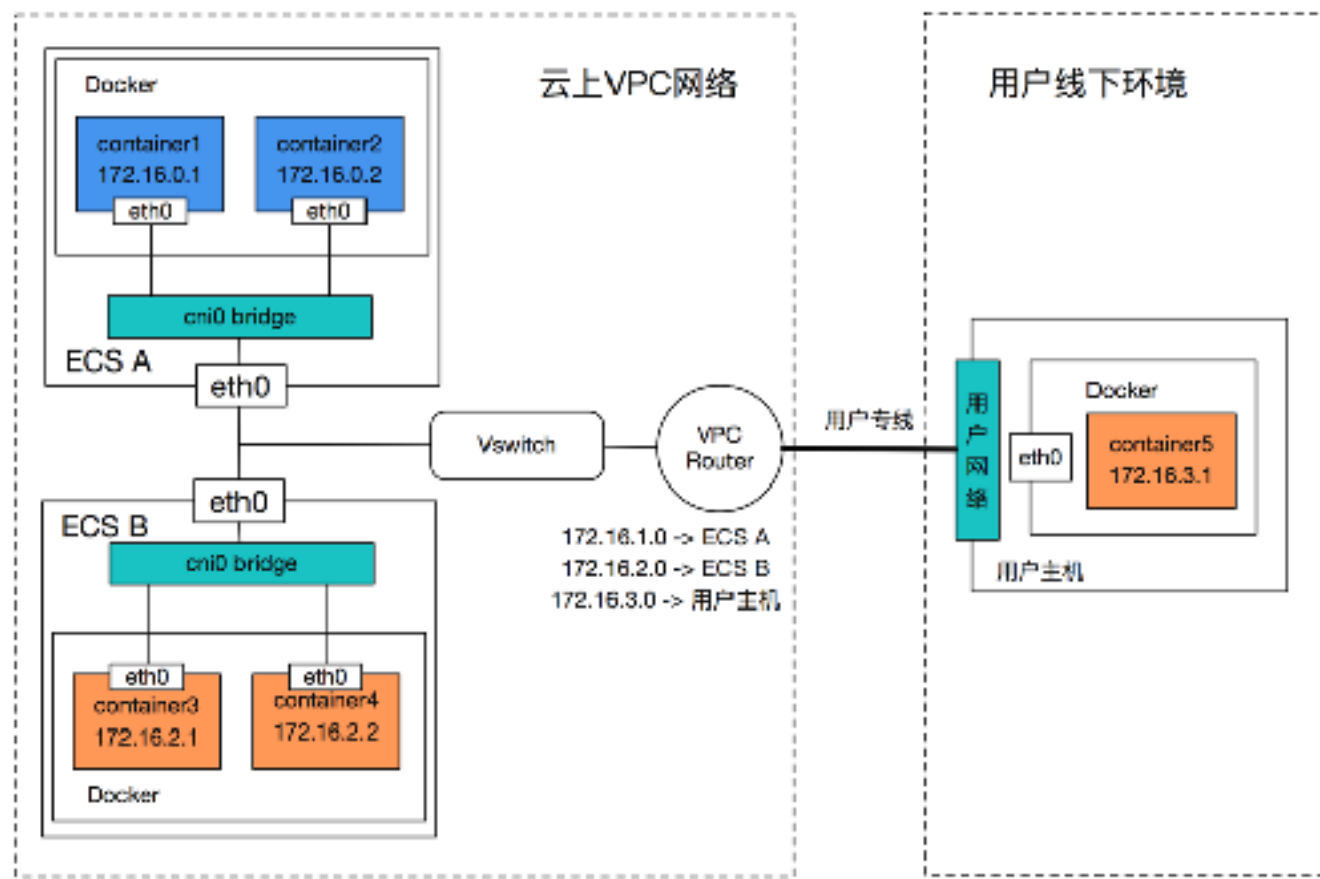
VPC：路由转发，去往172.16.0.0/24网段的数据包的下一跳是ECS1，性能最优的方案。

VXLAN：数据封包，overlay

Flannel alivpc driver

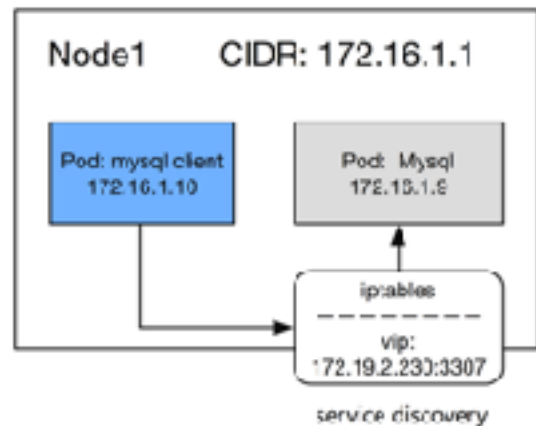
开源，连接容器网络

混合云支持



主办：

案例分析：一个神奇的内核网络配置问题



排查流程

kubectl get po 检查pod是否正常启动
kubectl get svc 检查服务是否创建
kubectl logs mysql 查看服务日志是否正常
netstat -anl | grep 3307 查看端口
iptables -t nat -vnL 查看数据流规则

.....

然而。。。一切看似正常

=> tcpdump

Client Pod: 172.16.1.10

```
root@172.16.1.10# tcpdump -i any -nnnn tcp and host 172.16.1.9 or host 172.16.1.10
08:35:55.731187 IP 172.16.1.10.49908 > 172.19.2.230.3307: Flags [S], seq 1112696560, win 292
08:35:55.731313 IP 172.16.1.9.80 > 172.16.1.10.49908: Flags [S.], seq 951763388, ack 1112696
08:35:55.731331 IP 172.16.1.10.49908 > 172.16.1.9.80: Flags [R], seq 1112696561, win 0, leng
08:35:56.732987 IP 172.16.1.10.49908 > 172.19.2.230.3307: Flags [S], seq 1112696560, win 292
08:35:56.733878 IP 172.16.1.9.80 > 172.16.1.10.49908: Flags [S.], seq 967416588, ack 1112696
08:35:56.733888 IP 172.16.1.10.49908 > 172.16.1.9.80: Flags [R], seq 1112696561, win 0, leng
```

On Mysql Server Pod 172.16.1.9

```
root@172.16.1.19# tcpdump -i any -nnnn tcp and host 172.16.1.9 or host 172.16.1.10
08:35:55.731249 IP 172.16.1.10.49908 > 172.16.1.9.80: Flags [S], seq 1112696560, win 29200,
08:35:55.731309 IP 172.16.1.9.80 > 172.16.1.10.49908: Flags [S.], seq 951763388, ack 1112696
08:35:55.731335 IP 172.16.1.10.49908 > 172.16.1.9.80: Flags [R], seq 1112696561, win 0, leng
08:35:56.733842 IP 172.16.1.10.49908 > 172.16.1.9.80: Flags [S], seq 1112696560, win 29200,
08:35:56.733872 IP 172.16.1.9.80 > 172.16.1.10.49908: Flags [S.], seq 967416588, ack 1112696
08:35:56.733892 IP 172.16.1.10.49908 > 172.16.1.9.80: Flags [R], seq 1112696561, win 0, leng
```

On Host 172.16.1.1

```
root@172.16.1.1# tcpdump -i any -nnnn tcp and host 172.16.1.9 or host 172.16.1.10
14:35:55.731285 IP 172.16.1.10.49908 > 172.19.2.230.3307: Flags [S], seq 1112696560, win 292
14:35:55.731285 IP 172.16.1.10.49908 > 172.19.2.230.3307: Flags [S], seq 1112696560, win 292
14:35:55.731244 IP 172.16.1.10.49908 > 172.16.1.9.80: Flags [S], seq 1112696560, win 29200,
14:35:55.731248 IP 172.16.1.10.49908 > 172.16.1.9.80: Flags [S], seq 1112696560, win 29200,
14:35:55.731277 ARP, Request who-has 172.16.1.10 tell 172.16.1.9, length 28
14:35:55.731281 ARP, Request who-has 172.16.1.10 tell 172.16.1.9, length 28
14:35:55.731283 ARP, Request who-has 172.16.1.10 tell 172.16.1.9, length 28
14:35:55.731285 ARP, Request who-has 172.16.1.10 tell 172.16.1.9, length 28
14:35:55.731288 ARP, Request who-has 172.16.1.10 tell 172.16.1.9, length 28
14:35:55.731277 ARP, Request who-has 172.16.1.10 tell 172.16.1.9, length 28
14:35:55.731300 ARP, Reply 172.16.1.10 is-at 0a:58:ac:10:01:0a, length 28
14:35:55.731307 ARP, Reply 172.16.1.10 is-at 0a:58:ac:10:01:0a, length 28
14:35:55.731310 IP 172.16.1.9.80 > 172.16.1.10.49908: Flags [S.], seq 951763388, ack 1112696
14:35:55.731312 IP 172.16.1.9.80 > 172.16.1.10.49908: Flags [S.], seq 951763388, ack 1112696
14:35:55.731332 IP 172.16.1.10.49908 > 172.16.1.9.80: Flags [R], seq 1112696561, win 0, leng
14:35:55.731334 IP 172.16.1.10.49908 > 172.16.1.9.80: Flags [R], seq 1112696561, win 0, leng
```

案例分析：一个神奇的内核网络配置问题

你会发现：

目的Pod返回给请求方的数据包里直接使用了自已的IP作为源IP。

DNAT规则没被iptables正确处理？

```
iptables

[root@iZ2ze8z3wmcY8sspd1l43rZ ~]# iptables -t nat -vnl
Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target    prot opt in     out     source            destination
 29237 1718K KUBE-SERVICES all  --  *      *      0.0.0.0/0         0.0.0.0/0
 4793  291K DOCKER all  --  *      *      0.0.0.0/0         0.0.0.0/0          ADC

Chain KUBE-SERVICES (2 references)
  pkts bytes target    prot opt in     out     source            destination
    0      0 KUBE-SVC-4N577FCL4MD7ZTDA tcp  --  *      *      0.0.0.0/0         0.0.0.0/0         172.19.2
    0      0 KUBE-NODEPORTS all  --  *      *      0.0.0.0/0         0.0.0.0/0

Chain KUBE-SVC-4N577FCL4MD7ZTDA (1 references)
  pkts bytes target    prot opt in     out     source            destination
    0      0 KUBE-SEP-NQHDPPG5W6CH56QM all  --  *      *      0.0.0.0/0         0.0.0.0/0         0.0.0.0/
    0      0 KUBE-SEP-5QE6BV4YQ5V2GPQM all  --  *      *      0.0.0.0/0         0.0.0.0/0         0.0.0.0/

Chain KUBE-SEP-5QE6BV4YQ5V2GPQM (1 references)
  pkts bytes target    prot opt in     out     source            destination
    0      0 KUBE-MARK-MASQ all  --  *      *      172.16.1.9        0.0.0.0/0
    0      0 DNAT      tcp  --  *      *      0.0.0.0/0         0.0.0.0/0          /*
```

====>

Linux 网桥的一项内核参数控制流经网桥的数据包是否会被iptables规则进一步处理。

net.bridge.bridge-nf-call-iptables

主办：



案例分析：Docker 的默认网络转发策略

Forward Table

```
[root@izbp138x ~]# iptables -vnL
Chain INPUT [policy ACCEPT 1988 packets, 742K bytes]
  pkts bytes target     prot opt in     out     source                 destination
  92M   32G KUBE-SERVICES all -- *      *       0.0.0.0/0             0.0.0.0/0
  92M   32G KUBE-FIREWALL all -- *      *       0.0.0.0/0             0.0.0.0/0

Chain FORWARD [policy DROP 0 packets, 0 bytes]
  pkts bytes target     prot opt in     out     source                 destination
   0     0 DOCKER-ISOLATION all -- *      *       0.0.0.0/0             0.0.0.0/0
   0     0 DOCKER      all -- *      docker0 0.0.0.0/0             0.0.0.0/0
   0     0 ACCEPT      all -- *      docker0 0.0.0.0/0             0.0.0.0/0
   0     0 ACCEPT      all -- docker0 !docker0 0.0.0.0/0             0.0.0.0/0
   0     0 ACCEPT      all -- docker0 docker0   0.0.0.0/0             0.0.0.0/0
```

iptables -P FORWARD ACCEPT

主办：



阿里云负载均衡SLB集成

LoadBalancer能力

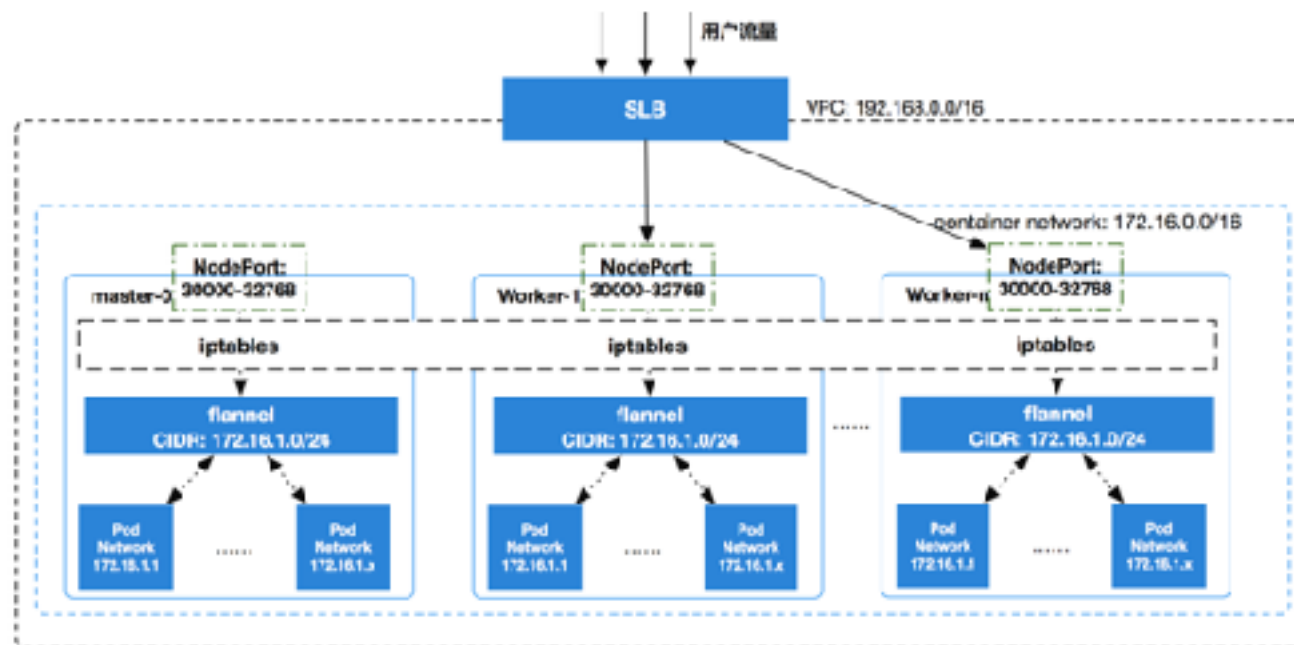
地址类型: internet/intranet

协议类型: tcp/udp/http/https

Region选择: 默认与自定义

收费类型: 按量付费/按带宽收费, 自定义带宽

健康检查: 自定义健康检查



主办:



案例分析：阿里云SLB 源IP保留

原则: 负载均衡是否能够保留客户请求的源IP取决于连接是否被SNAT过。

Type: ClusterIP

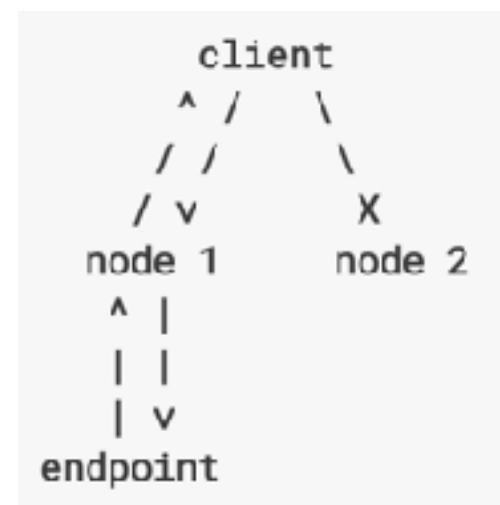
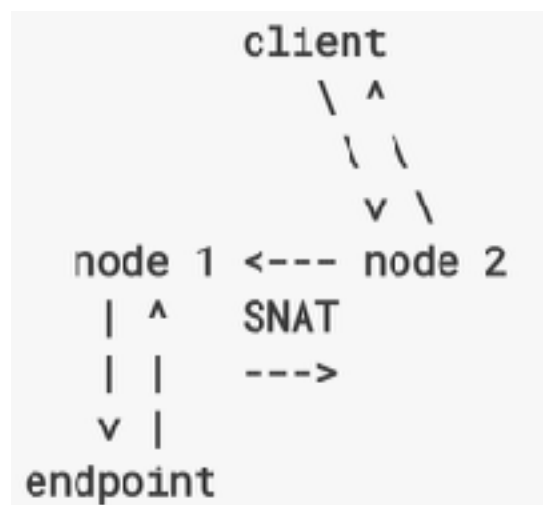
永远不会被SNAT，因此始终可以获得源IP

Type: NodePort

访问不在本节点上的Pod后端时会被SNAT。因此不一定能保留源IP

Type: LoadBalancer

依赖于NodePort



spec.externalTraffic: Local

主办:



小结

阿里云容器服务

阿里云Kubernetes集群

阿里云CloudProvider

Q&A

Thank you for your time

主办：

