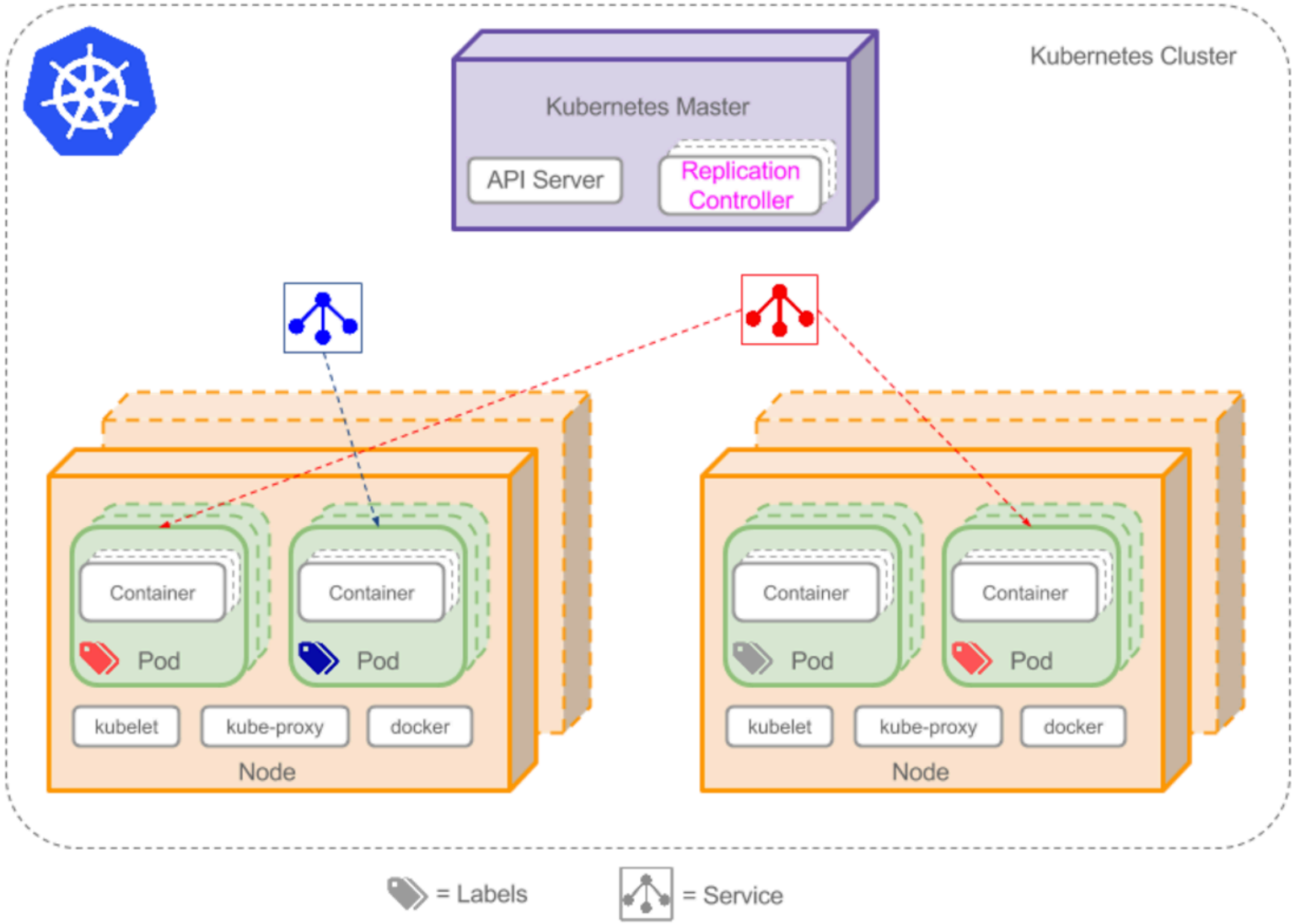
Kubernetes In Rancher Network

睿云智合 陈乐吉

目录

- Kubernetes 网络
- Rancher 网络
- Kubernetes In Rancher 网络
- 技术探索

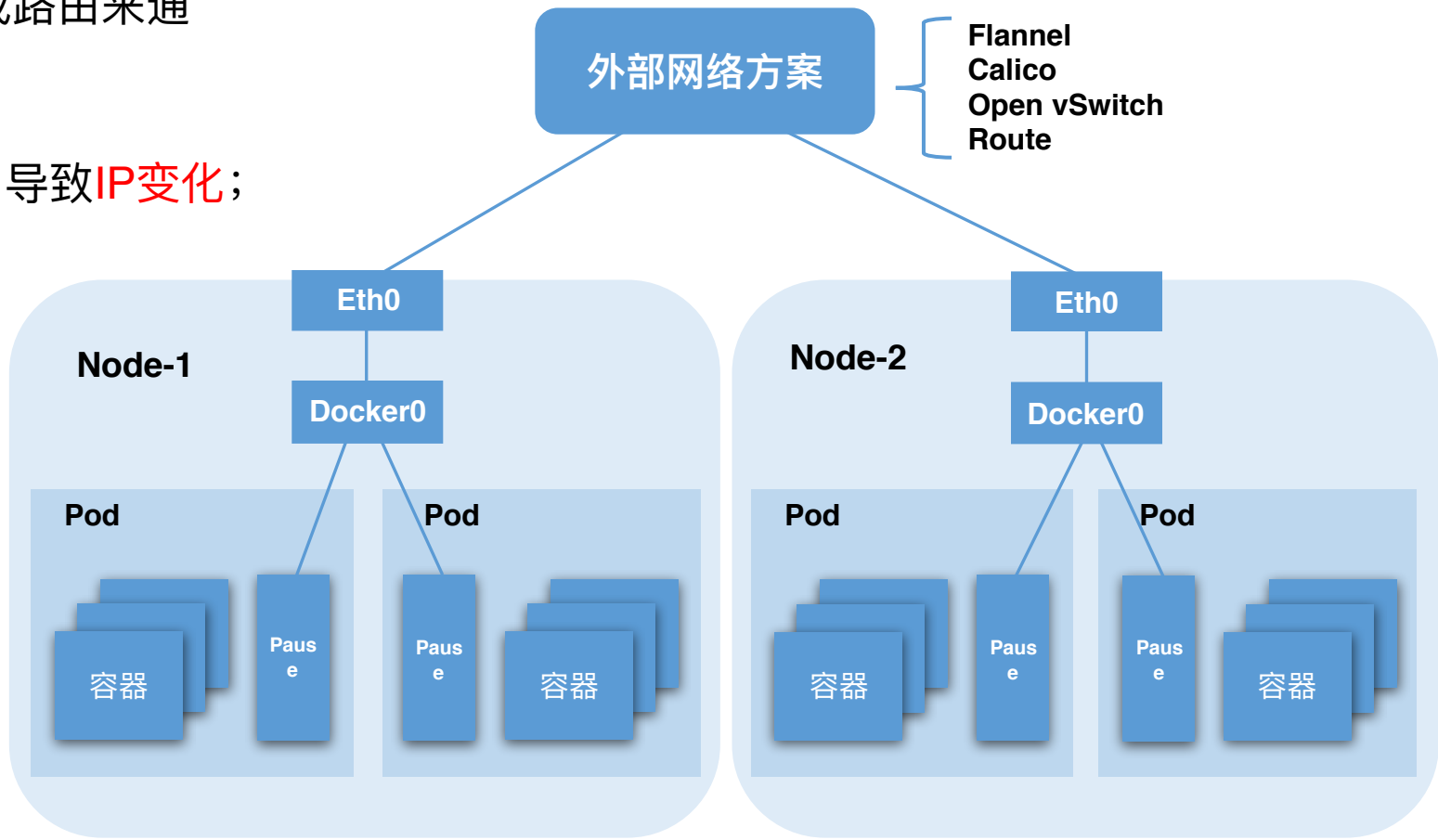
基本概念



Pod通信

- Pod内部: 端口互访;
- 同一Node上: 通过网桥通信;
- 不同Node上: 通过Overlay网络或路由来通信;

- RC管理的Pod的版本更新或重启导致IP变化;

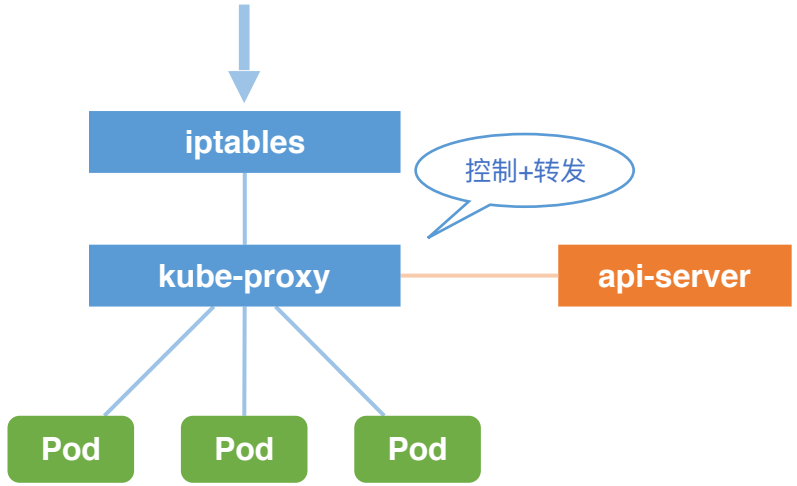


Service与Pod通信

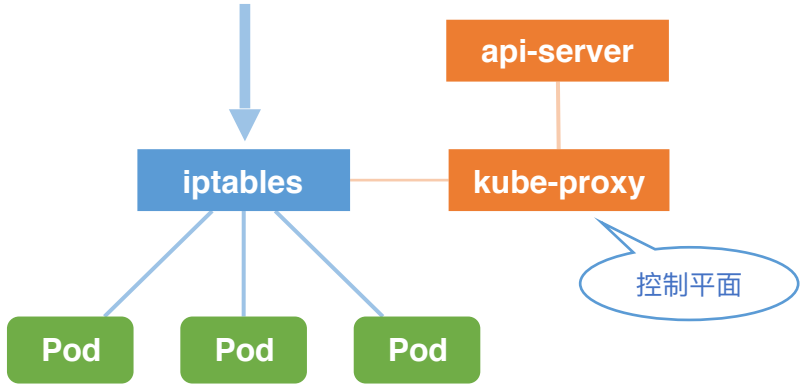
Service作为Pod的服务代理。

必须满足：

- 对外访问点(IP地址)固定，不轻易更新；
- 转发外部的访问请求到endpoints；
- 监控endpoints的变化，实时更新规则；
- 提供负载均衡。



Userspace模式



Iptables模式

Userspace模式

Cluster-IP类型：

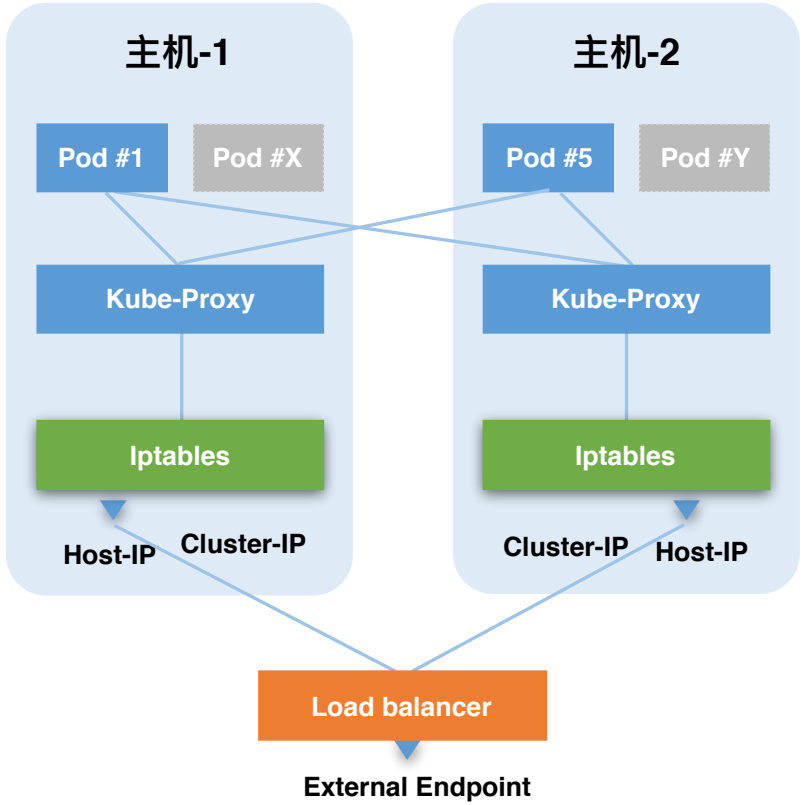
- Cluster-IP-range可配置；
- Cluster-IP不需要配置到网卡上；
- 报文被转到kube-proxy自动分配的端口上；
- 适用于Pod与Host访问Service；

Node-Port类型：

- 监听Node上所有IP的相应端口；
- 报文被转到kube-proxy自动分配的端口上；
- 适用于外部主机通过Node物理网络访问Service；

LoadBalancer类型：

- 外部LB的对应port监听请求，将流量转发到Node；
- 各Node上基于Node-Port的配置来接力转发；
- 适用于外网访问内部Service；



NAT 表	自定义链	备注
OUTPUT	KUBE-PORTALS-HOST	处理来自主机的流量
PREROUTING	KUBE-PORTALS-CONTAINER	处理来自容器的流量

Iptables模式

- 无论何类型均会分配到Cluster-IP;
- Cluster-IP的监听端口可指定;
- 对Iptables的使用，像openflow的流表逐级过滤;
- 需要对endpoint上发送来的流量做SNAT;
- 重点是如何做负载均衡。

NAT表标准链	自定义链		
PREROUTING	**KUBE-SERVICES** (All Traffic)	<1> **KUBE-SVC-<Service #1>** (match DIP/port = Cluster IP & port)	**KUBE-SEP-#1**
			KUBE-SEP-#2
OUTPUT		<2> **KUBE-SVC-<Service #2>** (match DIP/port = Cluster IP & port)	**KUBE-SEP-#3**
			KUBE-SEP-#4
		<3> **KUBE-NODEPORTS** (match other traffic)	**KUBE-SVC-<Service #3>** (match port = port)
POSTROUTING	**KUBE-POSTROUTING**	**MASQUERADE** (match 0x4000/0x4000)	

-A KUBE-SVC-GKN7Y2BSGW4NJTYL -m comment --comment "default/nginx-service:" -m statistic --mode random --probability 0.50000000000 -j KUBE-SEP-JS3AWPQDQ7R3OVVU

statistic

This module matches packets based on some statistic condition.

Supported options:

--mode mode
Set the matching mode of the matching rule, supported modes are random and nth.
--probability p
Set the probability for a packet to be randomly matched. It only works with the **random** mode. p must be within 0.0 and 1.0. The supported granularity is in 1/2147483648th increments.

服务发现

Service环境变量：

{SVCNAME}_SERVICE_HOST
{SVCNAME}_SERVICE_PORT
{SVCNAME}_SERVICE_PORT_{PORTNAME}

Link环境变量：

<alias>_NAME
<name>_PORT_<port>_<protocol>
<name>_PORT_<port>_<protocol>_ADDR
<name>_PORT_<port>_<protocol>_PORT
<name>_PORT_<port>_<protocol>_PROTO

```
root@nginx-xxx-2741081029-62msx:/# env | grep KUBE
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT=tcp://10.43.0.1:443
KUBERNETES_SERVICE_PORT=443
KUBERNETES_SERVICE_HOST=10.43.0.1
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT_443_TCP_ADDR=10.43.0.1
KUBERNETES_PORT_443_TCP=tcp://10.43.0.1:443
```

缺陷：

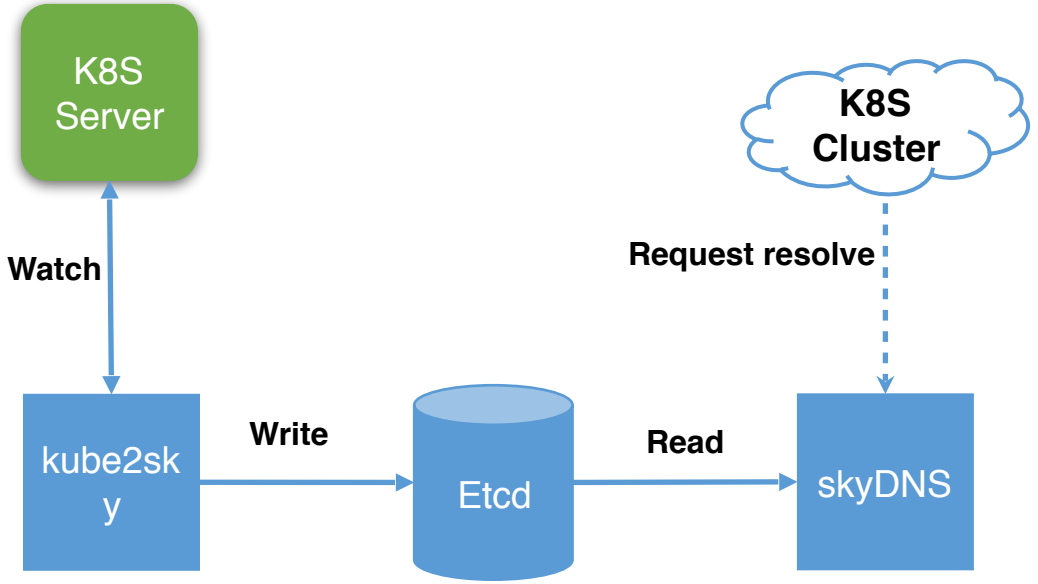
- Pod获取环境变量无法跨Namespace；
- 时序，Pod必须晚于Service创建；

Cluster DNS

- Kube2sky监听K8S Api-server；
- Service更新，kube2sky将记录保存到etcd；
- Skydns支持以etcd backend；
- Pod访问skyDNS解析域名。

<service_name>.<namespace_name>.<domain>
<service_name>.<namespace_name>.svc.<domain>

wise2c.com = wise2c.com.



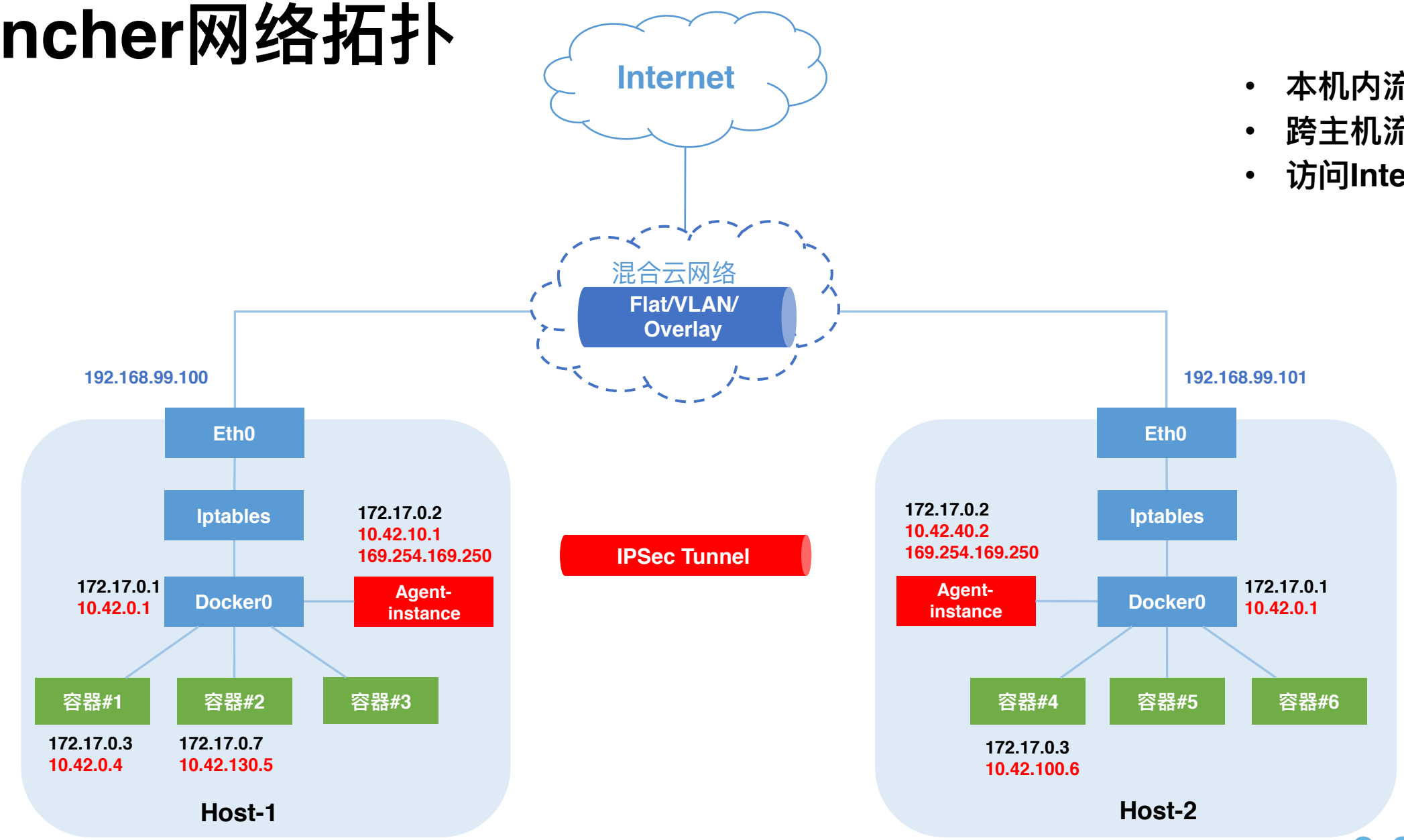
search default.svc.wise2c.com svc.wise2c.com wise2c.com
nameserver 192.168.99.1
options ndots:5

1. 绝对域名，即域名以`.`结尾, 仅查询该域名；
2. 相对域名，且域名包含的`.`的数目大于或等于option ndots命令指定的数，仅查询该域名；
3. 相对域名，且域名包含的`.`的数目少于option ndots命令指定的数，依次往传入的域名后追加search列表中的后缀；

目录

- Kubernetes 网络
- Rancher 网络
- Kubernetes In Rancher 网络
- 技术探索

Rancher网络拓扑



- 本机内流量
- 跨主机流量
- 访问Internet

IPSec Tunnel

隧道维护（基于config.json）

```
root@fa7f0b3c4057:/# ps -ef | grep rancher-net
root       816     1  0 Sep14 ?        00:01:51 /var/lib/cattle/bin/rancher-net
 --log /var/log/rancher-net.log -f /var/lib/cattle/etc/cattle/ipsec/config.json
 -c /var/lib/cattle/etc/cattle/ipsec -i 172.17.0.2/16 --pid-file /var/run/ranch
er-net.pid --gcm=true
```

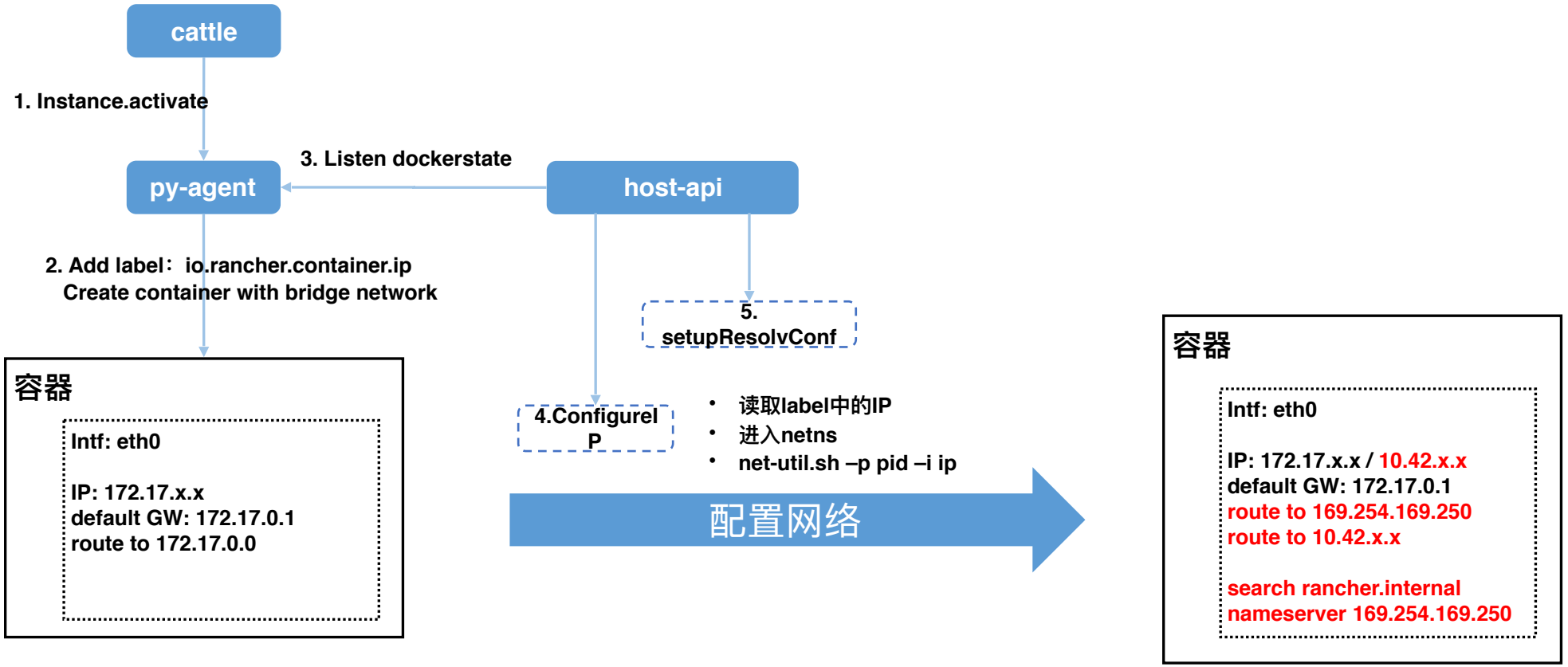
- 如果有新加主机，添加IPSec隧道；
- 如果有新加容器，但是已经存在IPSec隧道，更新xfrm policy；
- 如果有删除执行相反操作。

隧道路由：

- 容器内部协议栈判断到目的地址10.42.x.x 存在一条直连路由，发送ARP请求；
- 同一Host上的agent-instance容器监听ARP请求，接收报文后，判断该目的IP是否在本Host上；
- 对于目的IP不在该Host上的，使用自己的MAC响应ARP请求；
- 容器接收到ARP响应，发送业务报文到agent-instance；
- agent-instance容器内的IPSec policy将报文送入IPSec 隧道，发到目的Host的agent-instance做解包转发。

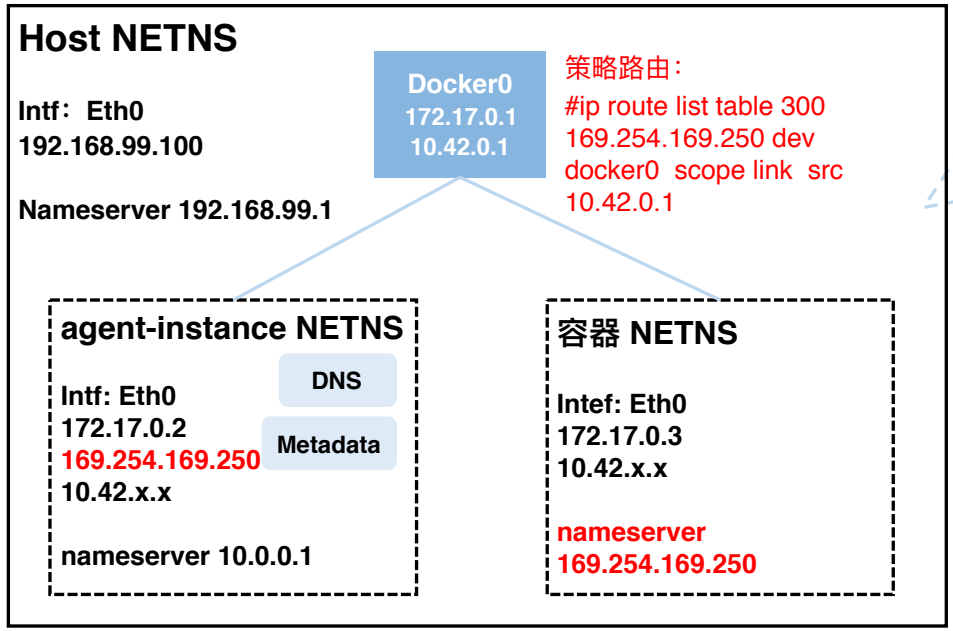
```
root@fa7f0b3c4057:/# cat /var/lib/cattle/etc/cattle/ipsec/config.json
{
"entries": [

{
    "peer": true,
    "ip": "10.42.220.166/16",
    "hostIp": "192.168.99.102"
},
{
    "ip": "10.42.127.203/16",
    "hostIp": "192.168.99.101"
},
{
    "peer": true,
    "self": true,
    "ip": "10.42.77.164/16",
    "hostIp": "192.168.99.101"
},
```

配置网络



关于地址<169.254.169.250>

- 被rancher-metadata和rancher-dns使用；
- 对外固定地址，访问不受网络因素影响；
- Rancher-DNS需要client IP。

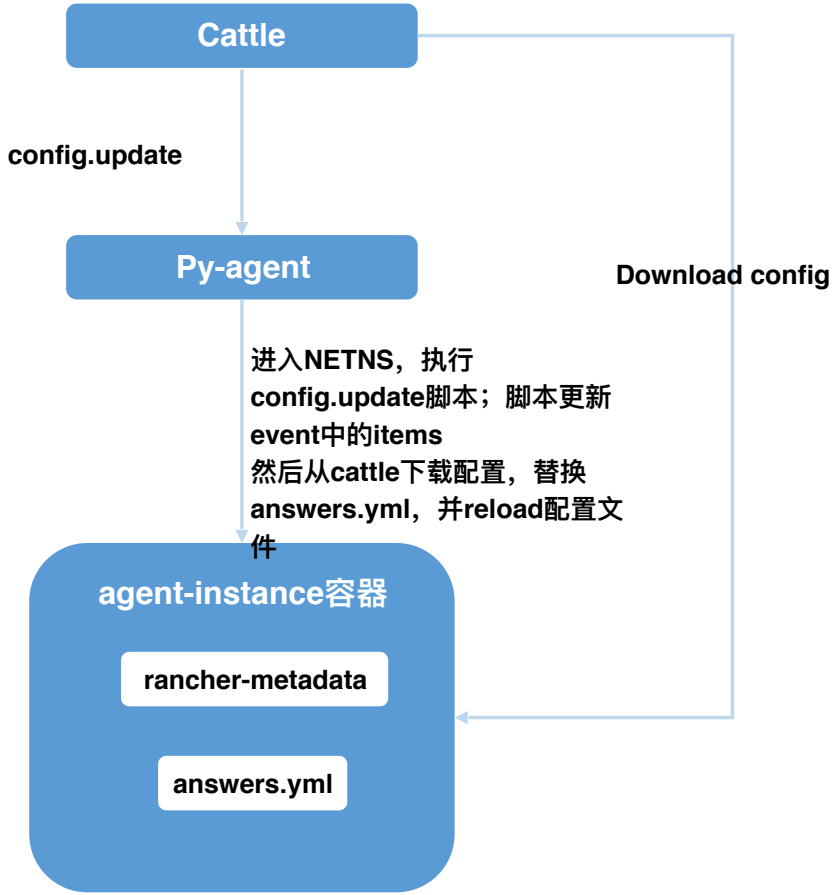


NAT表标准链	自定义链
PREROUTING	CATTLE_PREROUTING -A CATTLE_PREROUTING ! -s 10.42.0.0/16 -d 169.254.169.250/32 -m mac --mac-source 02:B9:48:71:98:75 -j MARK --set-xmark 0x1c5bc/0xffffffff
POSTROUTING	CATTLE_POSTROUTING -A CATTLE_POSTROUTING ! -s 10.42.0.0/16 -d 169.254.169.250/32 -m mark --mark 0x1c5bc -j SNAT --to-source 10.42.95.43

Rancher-Metadata

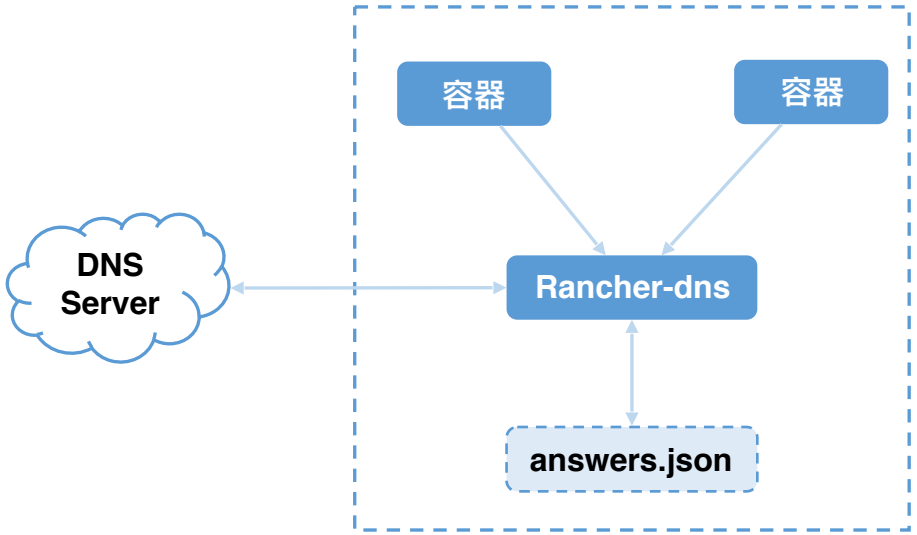
- 固定IP地址：169.254.169.250；
- Metadata Server：webserver，配置文件answers.yml；
- 支持reload：提供服务reload接口；
- 分布式：各host均存有metadata server；

```
root@f8e40d98bed5:/# ps -ef  | grep rancher-metadata
root       767     1  0 Sep05 ?        00:30:26 /var/lib/cattle/bin/rancher-metadata
-log /var/log/rancher-metadata.log -answers /var/lib/cattle/etc/cattle/metadata/answe
rs.yml -pid-file /var/run/rancher-metadata.pid
```



Rancher-DNS

- 分布式：每个Rancher-DNS只服务本Host上的容器；
- 源IP：记录按client_ip为key来存储;
- 两种特殊情况也会生成记录：
  1. 添加External-service：*<external_service>.<stack>.rancher.local*
  2. 为service添加别名：*<service_alias>.<stack>.rancher.local*

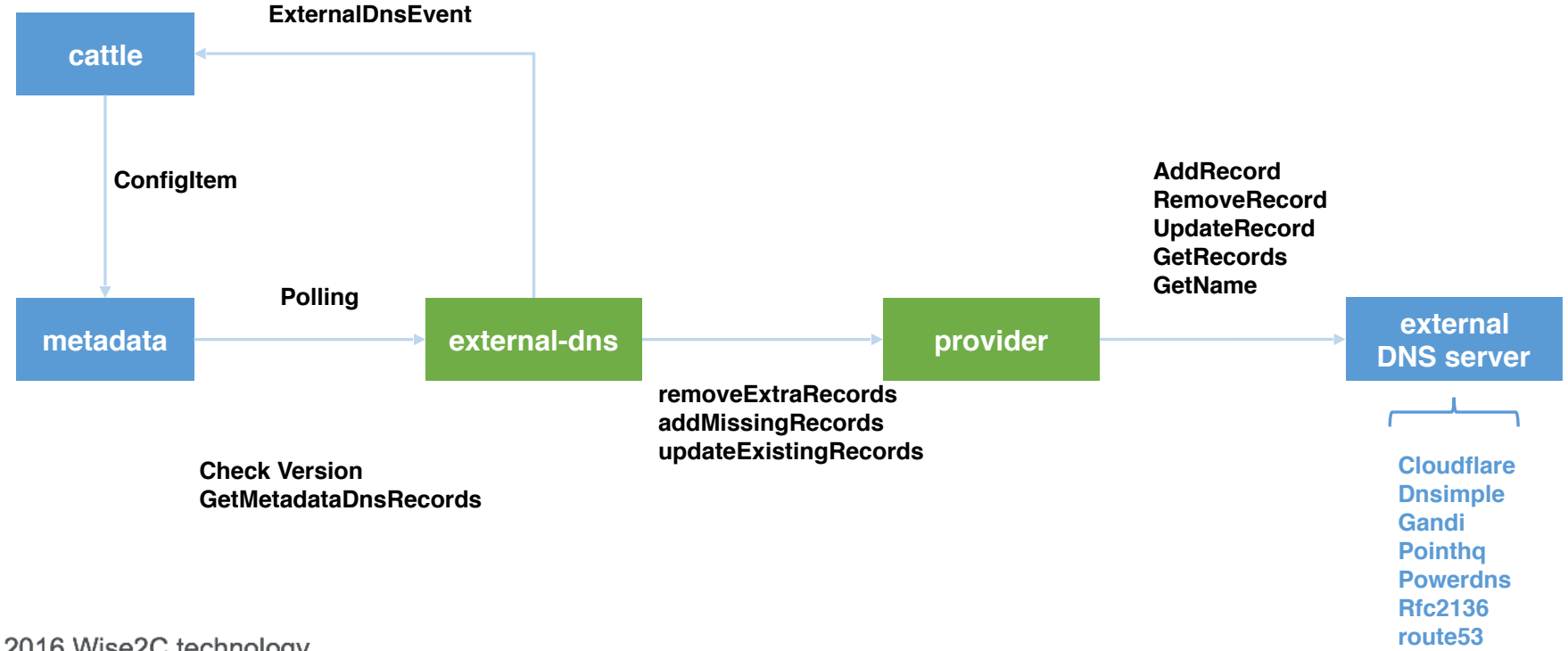



```
"10.42.229.51": {
  "search": [
    "wise2c-ci.rancher.internal",
    "wise2build-worker.wise2c-ci.rancher.internal",
    "rancher.internal"
  ],
  "recurse": □,
  "authoritative": □,
  "a": {
    "rabbitmq.wise2c-ci.rancher.internal.": {
      "answer": [
        "10.42.233.155"
      ]
    },
    "eureka-server.rancher.internal.": {
      "answer": [
        "10.42.36.223"
      ]
    },
    "rabbitmq.rancher.internal.": {
      "answer": [
        "10.42.233.155"
      ]
    },
    "eureka-server.wise2c-ci.rancher.internal.": {
      "answer": [
        "10.42.36.223"
      ]
    }
  },
  "cname": {}
},
```


```
root@f8e40d98bed5:/# ps -ef  | grep rancher-dns
root       854     1  0 Sep05 ?        00:07:21 /var/lib/cattle/bin/rancher-dns -
log /var/log/rancher-dns.log -answers /var/lib/cattle/etc/cattle/dns/answers.json
 -pid-file /var/run/rancher-dns.pid -ttl 1
```

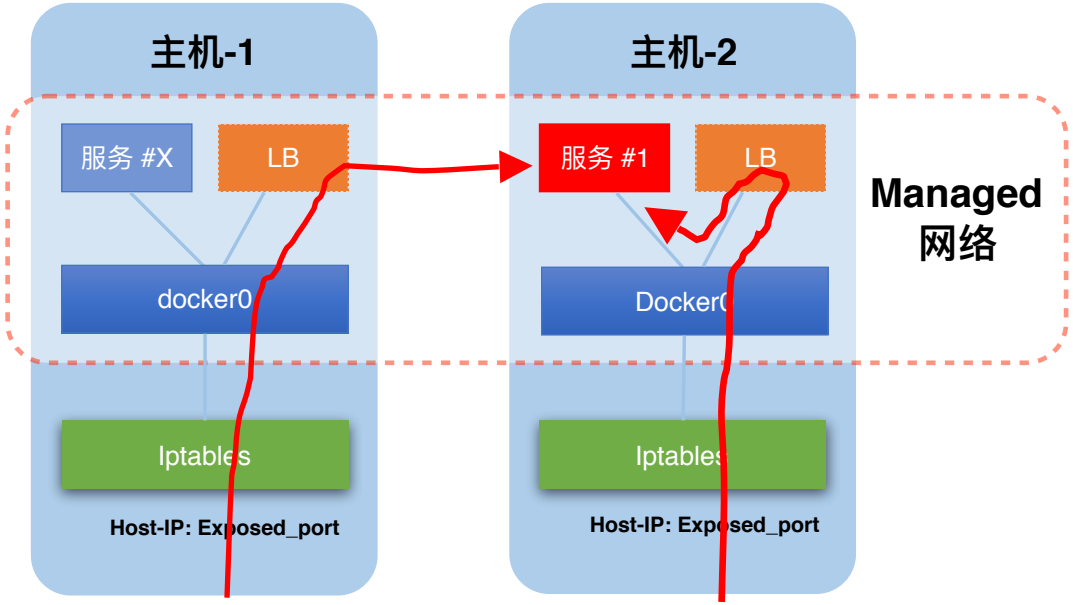
External-DNS

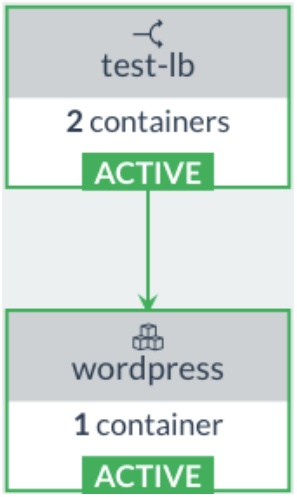
- 相当于DNS服务器的一个代理程序；
- Service必须Expose port到主机，否则无法生成域名记录；
- 需要为Host设置标签 io.rancher.host.external_dns_ip=*<IP_TO_BE_USED_FOR_EXTERNAL_DNS>*
- 域名规则：*<serviceName>.<stackName>.<environmentName>.<rootDomainName*



Load Balance

- 使用Haproxy做负载均衡；
- Lb通过managed网络转发流量到endpoints；
- LB端口expose到主机；
- 需要指定是否在每台主机上启动一个LB实例；

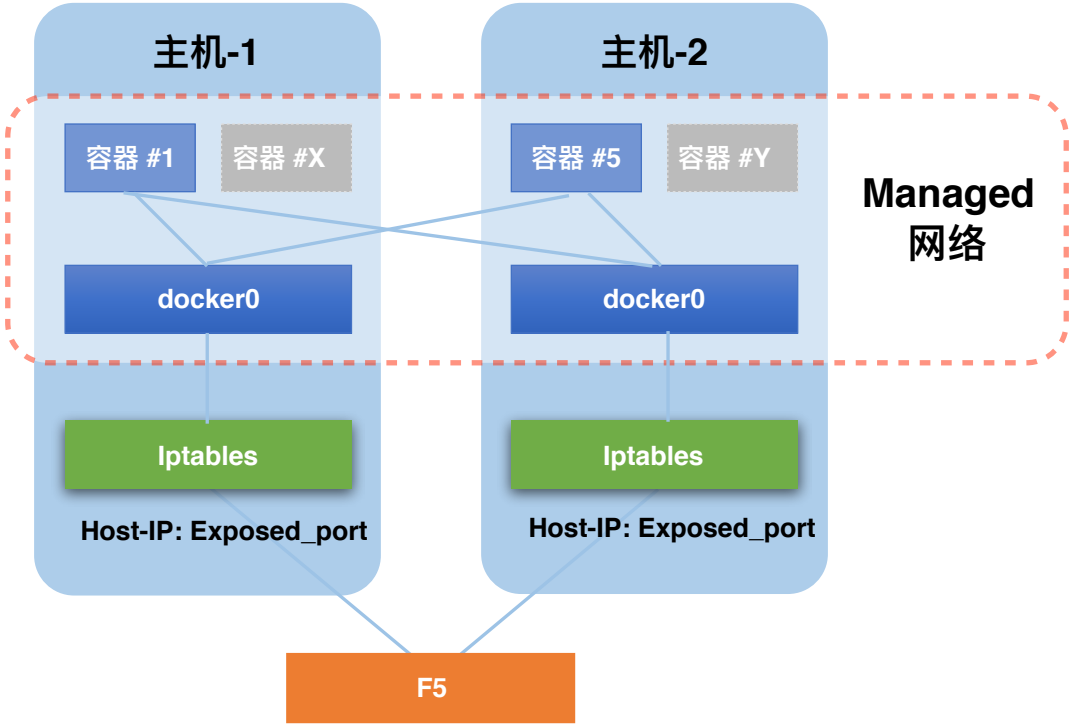





```
frontend 52339088-1944-4ca8-9880-e1352b49a64c_9090_frontend
        bind *:9090
        mode tcp

        default_backend 52339088-1944-4ca8-9880-e1352b49a64c_9090_0_backend

backend 52339088-1944-4ca8-9880-e1352b49a64c_9090_0_backend
        mode tcp
        server bc089070-44e7-457d-aa4e-a2d005dc3da8 10.42.138.191:80
        http-request set-header X-Forwarded-Port %[dst_port]
```

External-LB

- Service需expose port；
- io.rancher.service.external_lb_endpoint；
- External-LB自动创建Pool以及member信息，包含：<host_ip>:<exposed_port>



目录

- Kubernetes 网络
- Rancher 网络
- Kubernetes In Rancher 网络
- 技术探索

K8S In Rancher

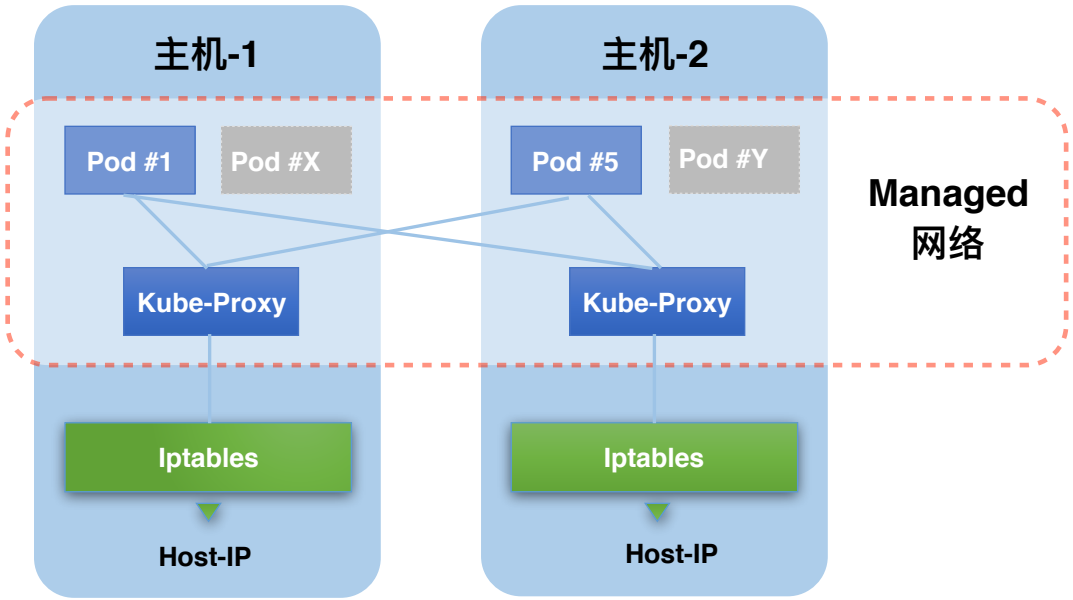
Rancher作为K8S的Cloud Provider:

- 一键部署K8S；
- 监控K8S各模块，提供自动恢复；
- 提供Pod互联支持；
- 提供DNS服务；
- 为K8S提供LB和ingress支持；

Rancher作为K8S的发行版本：

- 提供K8S部署catalog方式；
- 提供GUI；

Pod通信



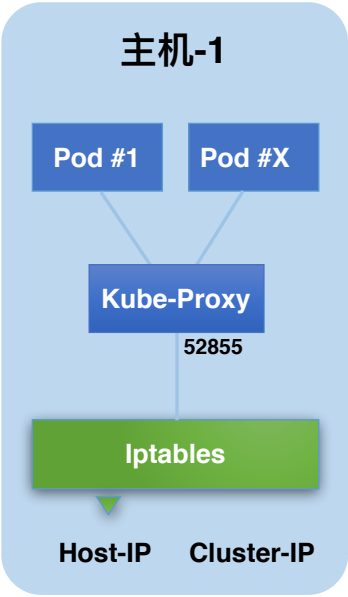
Service

NAT 表	自定义链	备注
OUTPUT	KUBE-PORTALS-HOST	来自主机流量
PREROUTING	KUBE-PORTALS-CONTAINER	来自容器流量

Cluster-IP类型：

- Cluster-IP-range固定： 10.43.x.x/16;

```
Chain KUBE-PORTALS-CONTAINER (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 REDIRECT   tcp  --  *      *       0.0.0.0/0            10.43.0.1            /* default/kubernetes:https */ tcp dpt:443 redir ports 34497
    0     0 REDIRECT   tcp  --  *      *       0.0.0.0/0            10.43.226.42         /* default/my-nginx: */ tcp dpt:8090 redir ports 52855

Chain KUBE-PORTALS-HOST (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DNAT       tcp  --  *      *       0.0.0.0/0            10.43.0.1            /* default/kubernetes:https */ tcp dpt:443 to:10.0.2.15:34497
    0     0 DNAT       tcp  --  *      *       0.0.0.0/0            10.43.226.42         /* default/my-nginx: */ tcp dpt:8090 to:10.0.2.15:52855
```



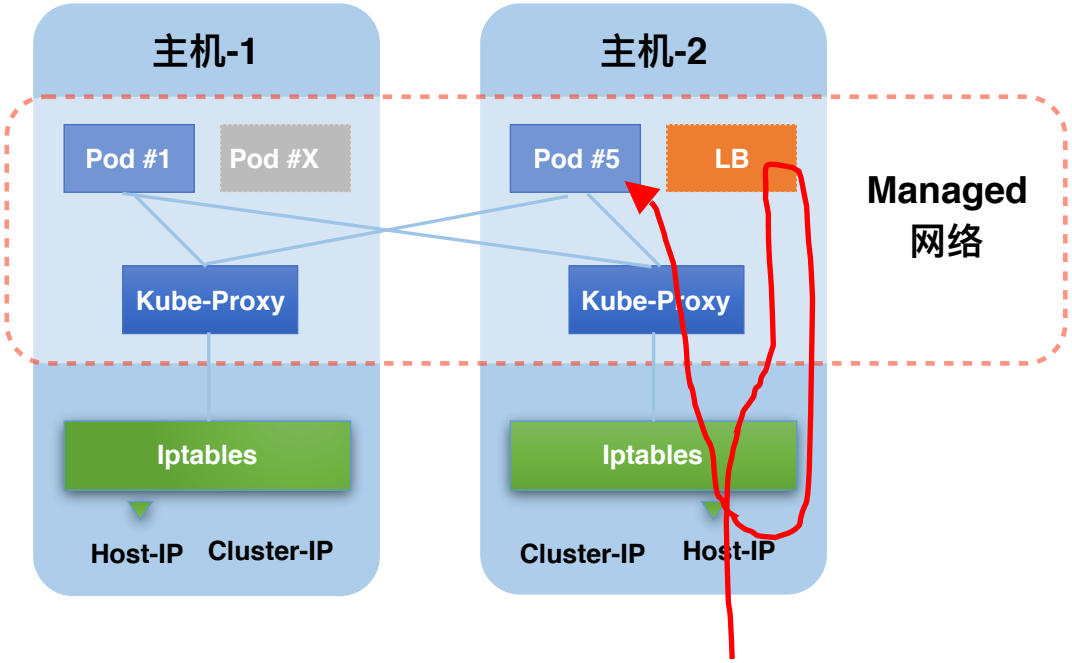
Node-Port类型：

```
Chain KUBE-NODEPORT-CONTAINER (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 REDIRECT   tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            /* default/my-nginx: */ tcp dpt:30612 redir ports 52855

Chain KUBE-NODEPORT-HOST (1 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DNAT       tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            /* default/my-nginx: */ tcp dpt:30612 to:10.0.2.15:52855
```

Service

LoadBalancer类型：

- 在Nodepot类型的基础上增加LB；
- 使用Rancher-LB，即haproxy作LB的instance；
- 流量可能两次经过主机协议栈。




```
frontend 07172fd3-49a7-4d56-acaf-375ae5d36891_80_frontend
        bind *:80
        mode tcp

        default_backend 07172fd3-49a7-4d56-acaf-375ae5d36891_80_0_backend

backend 07172fd3-49a7-4d56-acaf-375ae5d36891_80_0_backend
        mode tcp
        server e7892da4-630f-4c48-8f8b-5f494f752ba3 192.168.99.101:30448
        server cb0e6647-e68e-42f0-8ddd-dff15f27a7e6 192.168.99.102:30448
        http-request add-header X-Forwarded-Port %[dst_port]
```

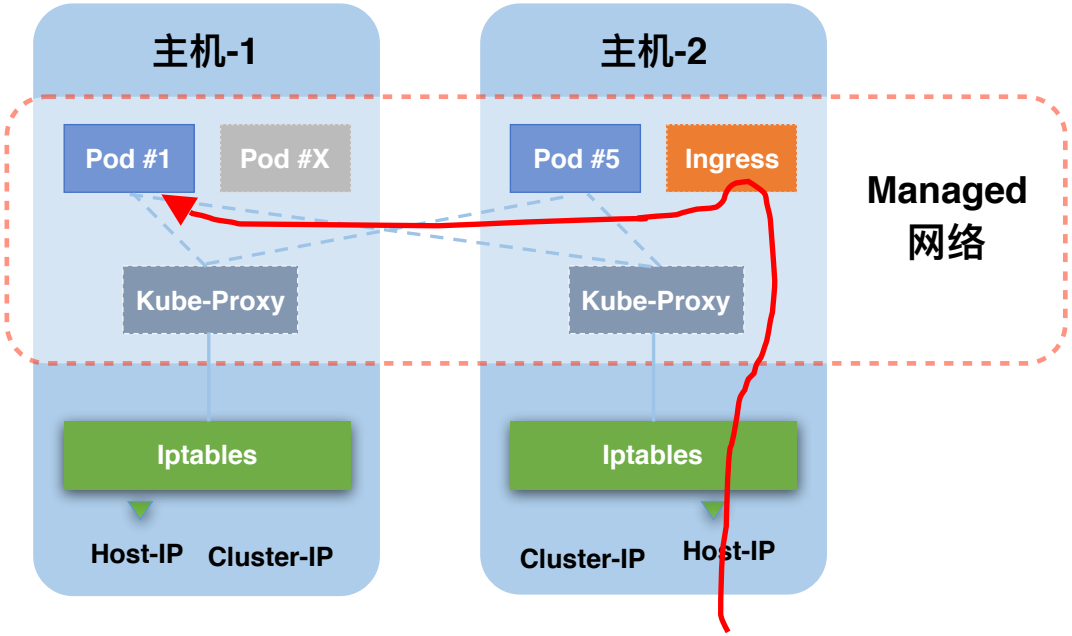
第一次经过port：80
第二次经过port：30448

Ingress

转发路径：

- 流量不经kube-proxy，直到endpoints；
- 只能处理HTTP和HTTPS；
- 支持认证；
- 支持基于HTTP的Host以及URL路径来转发。

```
frontend f2f4347e-e533-411c-8186-e2060ba6f543_80_frontend
        bind *:80
        mode http

                acl 0_host hdr(host) -i foo.com
                acl 0_host hdr(host) -i foo.com:80
        use_backend f2f4347e-e533-411c-8186-e2060ba6f543_80_0_backend if 0_host
                acl 1_host hdr(host) -i bar.com
                acl 1_host hdr(host) -i bar.com:80
        use_backend f2f4347e-e533-411c-8186-e2060ba6f543_80_1_backend if 1_host

backend f2f4347e-e533-411c-8186-e2060ba6f543_80_0_backend
        mode http
        server bbf565f7-8ce4-4401-a0fa-196725150441 10.42.247.160:9090
        server aad0754a-b75a-44d4-b5b3-f70988b64389 10.42.9.234:9090
        http-request add-header X-Forwarded-Port %[dst_port]

backend f2f4347e-e533-411c-8186-e2060ba6f543_80_1_backend
        mode http
        server c74b3259-e0c9-418c-a5a4-fc1ee65eb631 10.42.236.84:80
        server 176eed1c-cbd8-4486-b5a4-2a24515403f8 10.42.31.178:80
        server 78eac8f8-b532-4f20-ae56-452dc14de26c 10.42.83.240:80
        http-request add-header X-Forwarded-Port %[dst_port]
```



Ingress-controller

- **SyncQueue**

遍历LB controller中的所有LB的配置，然后将其应用到LBProvider中，最终实现从k8s中将LB配置应用到rancher的loadbalancer实例。

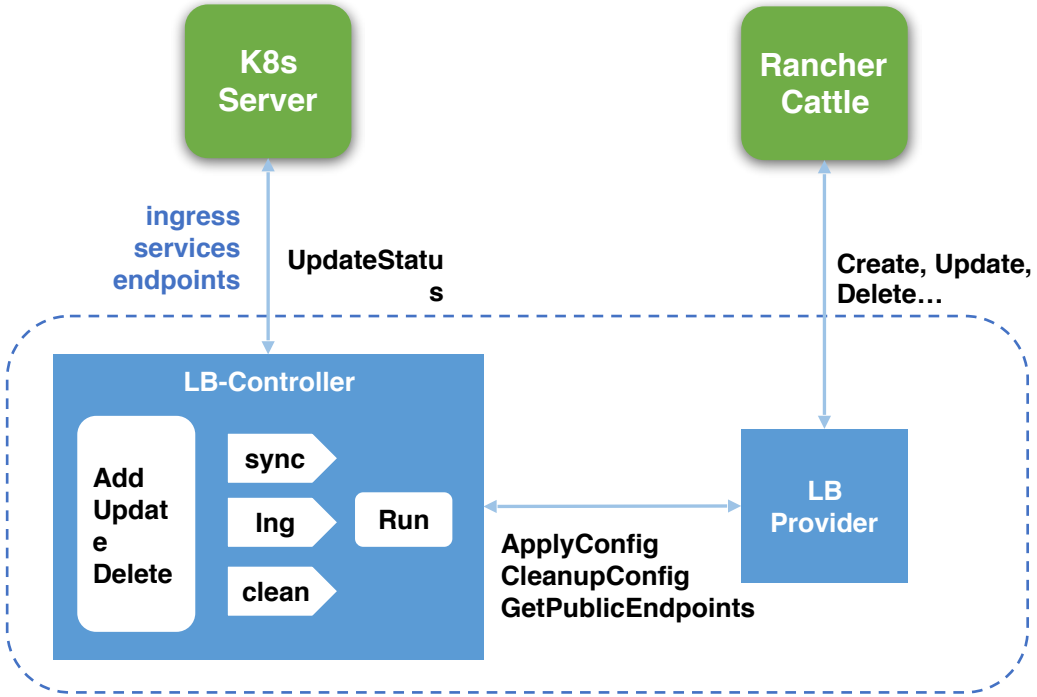
- **IngressQueue**

更新LB上的publicEndpoints地址，首先从rancher LB以及K8S中读取LB信息，然后以rancher LB的信息为准，一旦rancher LB的状态变化，就会调用K8S中的UpdaeStatus更新K8S中的LB状态。

- **CleanupQueue**

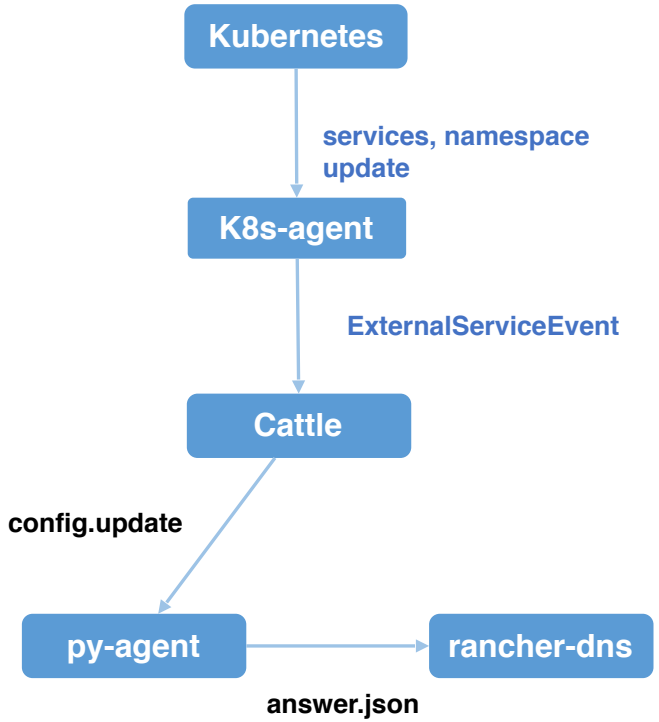
按照参数传入的key来调用rancher的API，清理LB配置。

	Ingress	Services	Endpoints
Sync Queue	• AddFunc • UpdateFunc	None	• AddFunc • UpdateFunc • DeleteFunc
Ingress Queue		None	None
Cleanup Queue	• DeleteFunc	None	None



服务发现

- 使用rancher-dns作为K8S服务发现DNS；
- 按照K8S的DNS规则，生成的DNS记录格式如下：
  <serviceName>.<namespaceName>.svc.cluster.local
- 同K8S一致，对于headless的service，service域名会对应到所有endpoints的多条A记录；



answer.json更新：

1. 由k8s-agent监听K8S的services和namespace事件；
2. K8s-agent检测到更新时，向cattle发送ExternalServiceEvent;
3. Cattle修改数据库并生成DNS记录，向所有的agent发送config.update事件；
4. 之后的流程同Rancher-DNS；

目录

- Kubernetes 网络
- Rancher 网络
- Kubernetes In Rancher 网络
- 技术探索

Pod网卡挂载 – CNI

/etc/cni/net.d/*

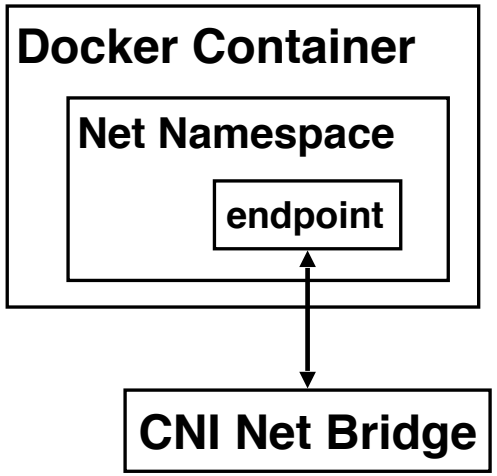
cniVersion
name
type
args
ipMasq
ipam
dns

```
{
    "name": "mynet",
    "type": "bridge",
    "bridge": "mynet0",
    "isDefaultGateway": true,
    "forceAddress": false,
    "ipMasq": true,
    "hairpinMode": true,
    "ipam": {
        "type": "host-local",
        "subnet": "10.10.0.0/16"
    }
}
```

2. netconf

1. 环境变量

CNI_VERSION
CNI_COMMAND
CNI_CONTAINERID
CNI_NETNS
CNI_IFNAME
CNI_ARGS
CNI_PATH

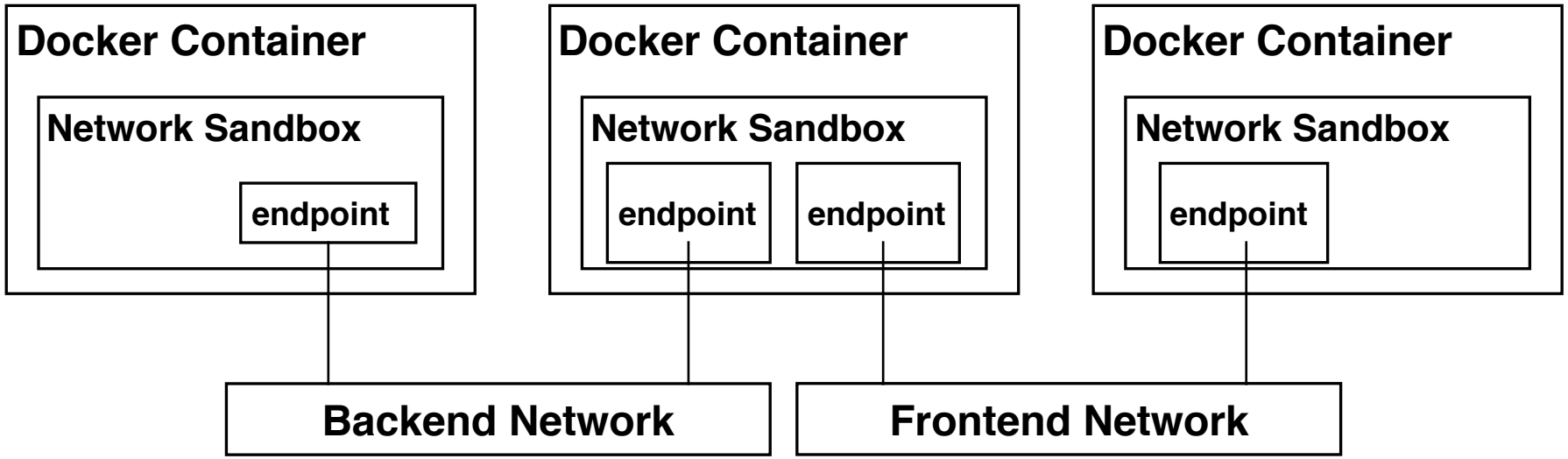


1. 创建网络为None的Pause容器；
2. 设置好环境变量，然后执行`plugin < netconf`；
3. plugin基于CNI参数获取命令、接口名和容器NETNS；
4. 基于netconf的配置创建CNI network bridge；
5. 进入容器NETNS，基于接口名创建endpoint；
6. 连接endpoint与CNI network bridge；
7. 基于ipam配置调用ipam插件获得IP地址，更新网络配置。

Docker网卡挂载 – CNM

- 使用方便，只需创建network的命令中指定driver；
- 需要在模块内实现libnetwork定义的接口；
- Docker-Engine通过REST-API或者Unix-socket方式访问接口；
- 由Docker-Engine触发对各接口的调用，能够自动释放资源；
- 通过IPAM分离出IP地址管理模块；

操作	接口
Create Network	• /NetworkDriver.CreateNetwork
Connect container to network	• /NetworkDriver.CreateEndpoint • /NetworkDriver.Join
Disconnect container from network	• /NetworkDriver.Leave • /NetworkDriver.DeleteEndpoint
Delete Network	• /NetworkDriver.DeleteNetwork



CNI 总结

优点：

- 将容器网卡向外的网络剥离出来，方便由专业人员来设计和开发网络；
- 接口更开放，netconf的args允许自定义，环境变量参数中，CNI_ARGS也提供了可扩展性；
- 只有ADD和DEL接口，相对于CNM来说，实现更单一，不用考虑多个操作之间时序问题；
- 调用的触发动作和时机均由第三方来控制，这为用户提供了更多的自主性；
- 还可以在plugin中实现更多的操作，几乎所有在容器network namespace中的操作均能在plugin中实现；

缺点：

- 对网络的操作，无法写入到docker daemon的配置中，对于故障排查会比较麻烦；
- 触发点不在docker daemon内，必须在外部集成第三方容器管理工具调用plugin来实现对网络的控制；
- 各主机上均需要netconf，同一个network的配置在各个host上均要求一致；如果有多个网络，那么同样要求配置内有多个网络的信息，且多主机之间的一致性；

CNI支持

cniglue（https://github.com/rancher/cniglue）

- 从环境变量读取“DOCKER_HOST_CONFIG”和“DOCKER_CONFIG”，从标准输入读入容器信息；
- 自动从读取的数据中提取出需要操作的container ID， NETNS, Intef_name等信息；
- 基于容器的host_config中network模式来读取/etc/docker/cni/<network>.d目录下的netconf配置文件；
- 可以将第三方CNI放到“/var/lib/cni/bin”, “/usr/local/sbin”,“/usr/sbin”,“/sbin”,“/usr/local/bin”,“/usr/bin”,“/bin”下；
- 遍历所有netconf配置文件，并按照netconf中配置的CNI type来调用CNI执行添加或者删除接口。

rancher-cni-ipam (https://github.com/rancher/rancher-cni-ipam)

- 使用标准的CNI IPAM，对CMD_ADD的处理是通过基于container-id去rancher-metadata查询由cattle分配的IP地址来实现；
- 对CMD_DEL的不再做任何处理，因为删除容器后cattle会自动释放IP地址。

如何与当前的Rancher集成？？

- 监听docker event，由start event触发对cniglue的调用?

VXLAN隧道支持

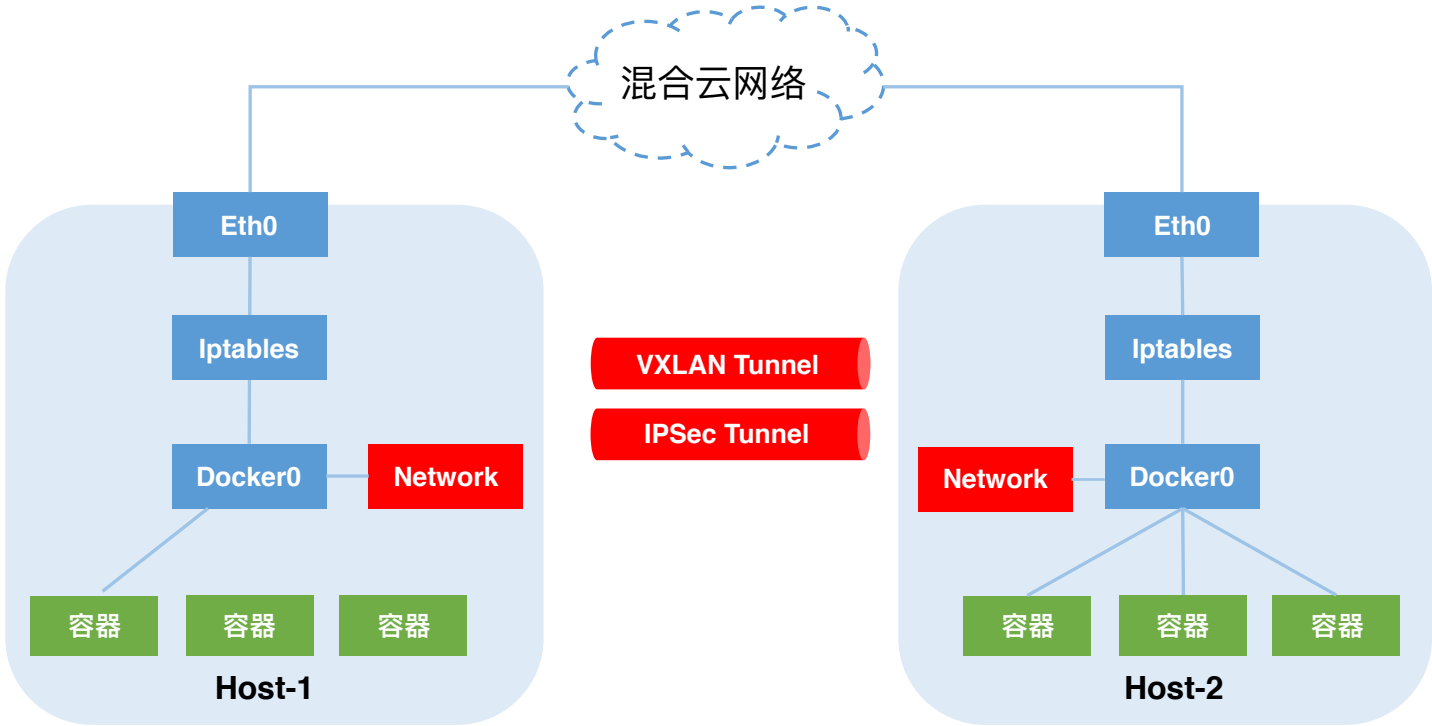
VXLAN tunnel （https://github.com/rancher/rancher-net/pull/16）

优势：

- 使用Linux kernel自带的vxlan模块，不用安装别的插件；
- 提供一种隧道，用于在混合云中通信，保留overlay的优势；
- 不使用Ipsec加密，提高传输效率；

缺点：

- 未使用VXLAN提供的L2 VPN特征；
- 未使用VXLAN支持2的24次方个VNI的特征，只使用了一个默认VNI（1024）；



Thanks!

微信号： chenleji