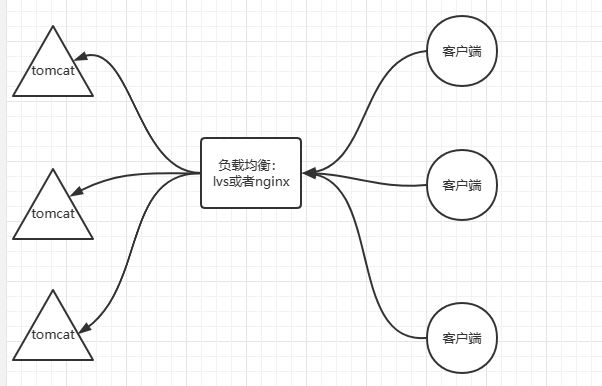
## Ingress和 Ingress Controller概述

## 1. 回顾下service四层负载



在k8s中为什么要做负载均衡？

Pod 漂移问题，可以理解成Pod IP是变化的

Kubernetes具有强大的副本控制能力，能保证在任意副本（Pod）挂掉时自动从其他机器启动一个新的，还可以动态扩容等。通俗地说，这个Pod可能在任何时刻出现在任何节点上，也可能在任何时刻死在任何节点上；那么自然随着Pod的创建和销毁，Pod IP 肯定会动态变化；那么如何把这个动态的Pod IP暴露出去？这里借助于Kubernetes的 Service 机制，Service可以以标签的形式选定一组带有指定标签的Pod，并监控和自动负载他们的Pod IP，那么我们向外暴露只暴露Service IP就行了；这就是NodePort模式：即在每个节点上开起一个端口，然后转发到内部Pod IP 上。

Service可以通过标签选择器找到它所关联的Pod。但是属于四层代理，只能基于IP和端口代理。

Service存在一个问题，什么问题呢？

Service的type类型有很多，如NodePort、clusterIp、loadbalancer、externalname，如果Service想要被k8s集群外部访问，需要用NodePort类型，但是NodePort类型的svc有如下几个问题：

nodeport会在物理机映射一个端口，绑定到物理机上，这样就导致，每个服务都要映射一个端口，端口过多，维护困难，还有就是Service底层使用的是iptables或者ipvs，仅支持四层代理，无法基于https协议做代理，因此我们这节课讲解的是ingress-nginx-controller七层代理。

四层负载和七层负载的区别：

区别：

1）四层负载：四层的负载均衡就是基于IP+端口的负载均衡：在三层负载均衡的基础上，通过发布三层的IP地址（VIP），然后加四层的端口号，来决定哪些流量需要做负载均衡，对需要处理的流量进行NAT处理，转发至后台服务器，并记录下这个TCP或者UDP的流量是由哪台服务器处理的，后续这个连接的所有流量都同样转发到同一台服务器处理。

2）七层的负载均衡就是基于虚拟的URL或主机IP的负载均衡：在四层负载均衡的基础上（没有四层是绝对不可能有七层的），再考虑应用层的特征，比如同一个Web服务器的负载均衡，除了根据VIP加80端口辨别是否需要处理的流量，还可根据七层的URL、浏览器类别、语言来决定是否要进行负载均衡。举个例子，如果你的Web服务器分成两组，一组是中文语言的，一组是英文语言的，那么七层负载均衡就可以当用户来访问你的域名时，自动辨别用户语言，然后选择对应的语言服务器组进行负载均衡处理。

3）四层负载均衡工作在传输层，七层负载均衡工作在应用层

OSI七层模型：



## 2. Ingress介绍

Ingress官网定义：Ingress可以把进入到集群内部的请求转发到集群中的一些服务上，从而可以把服务映射到集群外部。Ingress 能把集群内Service 配置成外网能够访问的 URL，流量负载均衡，提供基于域名访问的虚拟主机等。

Ingress简单的理解就是你原来需要改Nginx配置，然后配置各种域名对应哪个 Service，现在把这个动作抽象出来，变成一个 Ingress 对象，你可以用 yaml 创建，每次不要去改Nginx 了，直接改yaml然后创建/更新就行了；那么问题来了：”Nginx 该怎么处理？”

Ingress总结：ingress是k8s中的资源，主要是管理ingress-controller这个代理的配置文件

Ingress Controller 这东西就是解决 “Nginx 的处理方式” 的；Ingress Controller 通过与 Kubernetes API 交互，动态的去感知集群中Ingress规则变化，然后读取他，按照他自己模板生成一段 Nginx 配置，再写到 Nginx Pod 里，最后 reload 一下

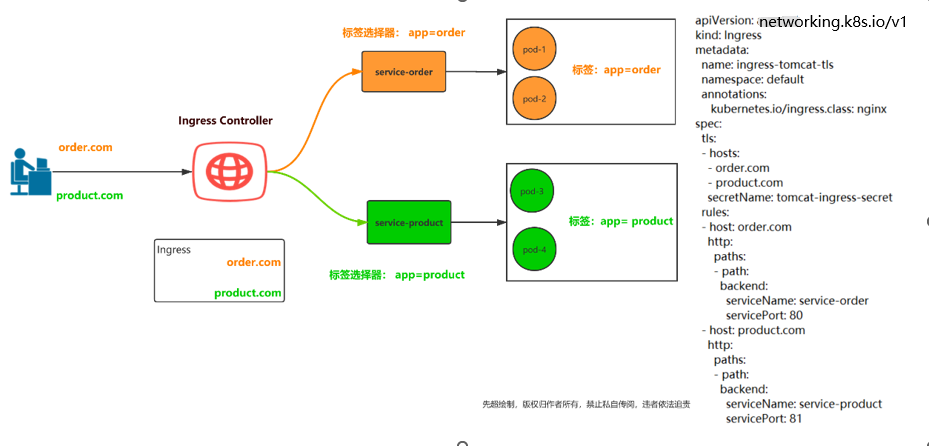
## 3.Ingress Controller介绍

Ingress Controller是一个七层负载均衡调度器，客户端的请求先到达这个七层负载均衡调度器，由七层负载均衡器在反向代理到后端pod，常见的七层负载均衡器有nginx、traefik，以我们熟悉的nginx为例，假如请求到达nginx，会通过upstream反向代理到后端pod应用，但是后端pod的ip地址是一直在变化的，因此在后端pod前需要加一个service，这个service只是起到分组的作用，那么我们upstream只需要填写service地址即可。

Ingress-controller里面封装就是nginx，有的同学问？

我直接在物理机上装个nginx就行了啊，为啥还弄个Ingress-controller？

Nginx：nginx配置文件一改动，你还需要手动reload一下才可以生效，但是如果用ingress-controller封装的nginx，你ingress维护配置，ingress创建好之后，会自动的把配置文件传到ingress-controller这个pod里，会自动进行reload，然后配置就生效了。



## 4.Ingress和Ingress Controller总结

Ingress Controller

Ingress Controller结合Ingress 定义的规则生成配置，然后动态更新ingress-controller里的Nginx 或者trafik负载均衡器，并刷新使配置生效，来达到服务自动发现的作用。

Ingress 则是定义规则，通过它定义某个域名的请求过来之后转发到集群中指定的 Service。它可以通过 Yaml 文件定义，可以给一个或多个 Service 定义一个或多个 Ingress 规则。

## 5. 使用Ingress Controller代理k8s内部pod的流程

（1）部署Ingress controller，我们ingress controller使用的是nginx

（2）创建Pod应用，可以通过控制器创建pod

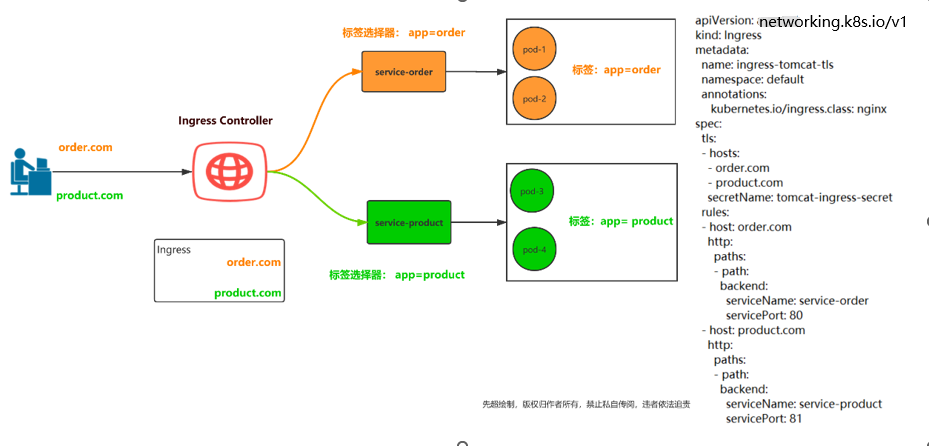
（3）创建Service，用来分组pod

（4）创建Ingress http，测试通过http访问应用

（5）创建Ingress https，测试通过https访问应用

客户端通过七层调度器访问后端pod的方式

使用七层负载均衡调度器ingress controller时，当客户端访问kubernetes集群内部的应用时，数据包走向如下图流程所示：



## 6、Ingress-controller高可用

****Ingress Controller是集群流量的接入层，对它做高可用非常重要，可以基于keepalive实现nginx-ingress-controller高可用，具体实现如下：****

Ingress-controller根据Deployment+ nodeSeletor+pod反亲和性方式部署在k8s指定的两个work节点，nginx-ingress-controller这个pod共享宿主机ip，然后通过keepalive+nginx实现nginx-ingress-controller高可用

参考：<https://github.com/kubernetes/ingress-nginx>

<https://github.com/kubernetes/ingress-nginx/tree/main/deploy/static/provider/baremetal>

特别注意：因为我们现在安装的k8s版本是1.25，那就需要按照文档步骤ctr -n=k8s.io images import导出镜像，如果k8s版本是1.24之前的，可以用docker load -i解压，视频里用的docker load -i，现在我们课程安装更新到了k8s1.25，所以导出镜像需要用ctr -n=k8s.io images import

[root@xianchaonode1 ~]# ctr -n=k8s.io images import ingress-nginx-controllerv1.1.0.tar.gz

[root@xianchaonode1 ~]# docker load -i ingress-nginx-controllerv1.1.0.tar.gz

[root@xianchaonode1 ~]# ctr -n=k8s.io images import kube-webhook-certgen-v1.1.0.tar.gz

[root@xianchaonode1 ~]#docker load -i kube-webhook-certgen-v1.1.0.tar.gz

[root@xianchaonode2 ~]# ctr -n=k8s.io images import ingress-nginx-controllerv1.1.0.tar.gz

[root@xianchaonode2 ~]# docker load -i ingress-nginx-controllerv1.1.0.tar.gz

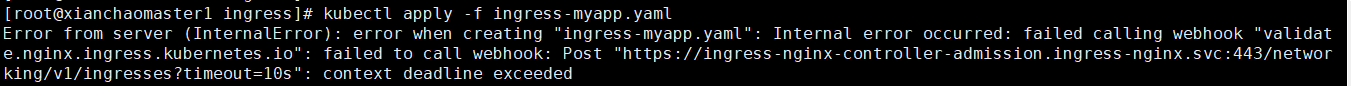
[root@xianchaonode2 ~]# ctr -n=k8s.io images import kube-webhook-certgen-v1.1.0.tar.gz

[root@xianchaonode2 ~]# docker load -i kube-webhook-certgen-v1.1.0.tar.gz

#更新yaml文件：

[root@xianchaomaster1 ~]# kubectl apply -f ingress-myapp.yaml

报错如下：



解决办法：

[root@xianchaomaster1 ]# kubectl delete -A ValidatingWebhookConfiguration ingress-nginx-admission

[root@xianchaomaster1 ~]# kubectl apply -f ingress-myapp.yaml

[root@xianchaomaster1 ~]# kubectl get pods -n ingress-nginx -o wide

NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES

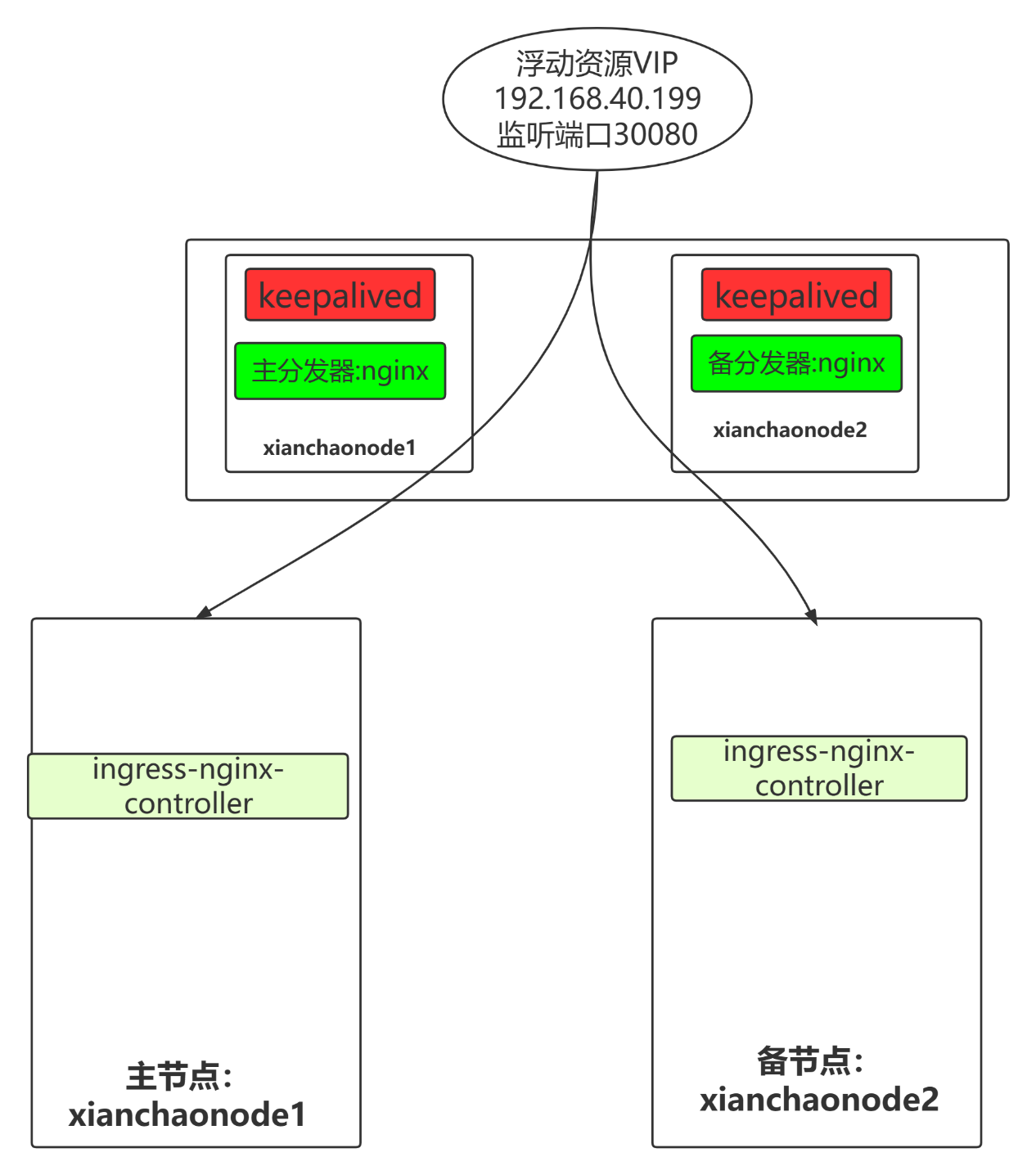
ingress-nginx-admission-create-k4fmn 0/1 Completed 0 103s 10.244.1.12 xianchaonode1 <none> <none>

ingress-nginx-admission-patch-x87h8 0/1 Completed 1 103s 10.244.1.11 xianchaonode1 <none> <none>

ingress-nginx-controller-6c8ffbbfcf-cjj9t 1/1 Running 0 103s 192.168.40.181 xianchaonode1 <none> <none>

ingress-nginx-controller-6c8ffbbfcf-wpt26 1/1 Running 0 103s 192.168.40.182 xianchaonode2 <none> <none>

### 10.3.1 通过keepalived+nginx实现ingress-nginx-controller高可用



1、安装nginx主备：

在xianchaonode1和xianchaonode2上做nginx主备安装

[root@xianchaonode1 ~]# yum install epel-release nginx keepalived nginx-mod-stream -y

[root@xianchaonode2 ~]# yum install epel-release nginx keepalived nginx-mod-stream -y

2、修改nginx配置文件。主备一样

[root@xianchaonode1 ~]# cat /etc/nginx/nginx.conf

user nginx;

worker\_processes auto;

error\_log /var/log/nginx/error.log;

pid /run/nginx.pid;

include /usr/share/nginx/modules/\*.conf;

events {

worker\_connections 1024;

}

# 四层负载均衡，为两台Master apiserver组件提供负载均衡

stream {

log\_format main '$remote\_addr $upstream\_addr - [$time\_local] $status $upstream\_bytes\_sent';

access\_log /var/log/nginx/k8s-access.log main;

upstream k8s-ingress-controller {

server 192.168.40.181:80 weight=5 max\_fails=3 fail\_timeout=30s; # xianchaonode1 IP:PORT

server 192.168.40.182:80 weight=5 max\_fails=3 fail\_timeout=30s; # xianchaonode2 IP:PORT

}

server {

listen 30080;

proxy\_pass k8s-ingress-controller;

}

}

http {

log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

'$status $body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

access\_log /var/log/nginx/access.log main;

sendfile on;

tcp\_nopush on;

tcp\_nodelay on;

keepalive\_timeout 65;

types\_hash\_max\_size 2048;

include /etc/nginx/mime.types;

default\_type application/octet-stream;

}

#备注：

#server 反向服务地址和端口

#weight 权重

#max\_fails 失败多少次，认为主机已挂掉，则踢出

#fail\_timeout 踢出后重新探测时间

[root@xianchaonode2 ~]# cat /etc/nginx/nginx.conf

user nginx;

worker\_processes auto;

error\_log /var/log/nginx/error.log;

pid /run/nginx.pid;

include /usr/share/nginx/modules/\*.conf;

events {

worker\_connections 1024;

}

# 四层负载均衡，为两台Master apiserver组件提供负载均衡

stream {

log\_format main '$remote\_addr $upstream\_addr - [$time\_local] $status $upstream\_bytes\_sent';

access\_log /var/log/nginx/k8s-access.log main;

upstream k8s-ingress-controller{

server 192.168.40.181:80 weight=5 max\_fails=3 fail\_timeout=30s; # xianchaonode1 IP:PORT

server 192.168.40.182:80 weight=5 max\_fails=3 fail\_timeout=30s; # xianchaonode2 IP:PORT

}

server {

listen 30080;

proxy\_pass k8s-ingress-controller;

}

}

http {

log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

'$status $body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

access\_log /var/log/nginx/access.log main;

sendfile on;

tcp\_nopush on;

tcp\_nodelay on;

keepalive\_timeout 65;

types\_hash\_max\_size 2048;

include /etc/nginx/mime.types;

default\_type application/octet-stream;

}

注意：nginx监听端口变成大于30000的端口，比方说30080,这样访问域名:30080就可以了，必须是满足大于30000以上，才能代理ingress-controller

3、keepalive配置

主keepalived

[root@xianchaonode1 ~]# cat /etc/keepalived/keepalived.conf

global\_defs {

notification\_email {

acassen@firewall.loc

failover@firewall.loc

sysadmin@firewall.loc

}

notification\_email\_from Alexandre.Cassen@firewall.loc

smtp\_server 127.0.0.1

smtp\_connect\_timeout 30

router\_id NGINX\_MASTER

}

vrrp\_script check\_nginx {

script "/etc/keepalived/check\_nginx.sh"

}

vrrp\_instance VI\_1 {

state MASTER

interface ens33 # 修改为实际网卡名

virtual\_router\_id 51 # VRRP 路由 ID实例，每个实例是唯一的

priority 100 # 优先级，备服务器设置 90

advert\_int 1 # 指定VRRP 心跳包通告间隔时间，默认1秒

authentication {

auth\_type PASS

auth\_pass 1111

}

# 虚拟IP

virtual\_ipaddress {

192.168.40.199/24

}

track\_script {

check\_nginx

}

}

#vrrp\_script：指定检查nginx工作状态脚本（根据nginx状态判断是否故障转移）

#virtual\_ipaddress：虚拟IP（VIP）

[root@xianchaonode1 ~]# cat /etc/keepalived/check\_nginx.sh

#!/bin/bash

#1、判断Nginx是否存活

counter=$(ps -ef |grep nginx | grep sbin | egrep -cv "grep|$$" )

if [ $counter -eq 0 ]; then

    #2、如果不存活则尝试启动Nginx

    service nginx start

    sleep 2

    #3、等待2秒后再次获取一次Nginx状态

    counter=$(ps -ef |grep nginx | grep sbin | egrep -cv "grep|$$" )

    #4、再次进行判断，如Nginx还不存活则停止Keepalived，让地址进行漂移

    if [ $counter -eq 0 ]; then

        service  keepalived stop

    fi

fi

[root@xianchaonode1 ~]# chmod +x /etc/keepalived/check\_nginx.sh

备keepalive

[root@xianchaonode2 ~]# cat /etc/keepalived/keepalived.conf

global\_defs {

notification\_email {

acassen@firewall.loc

failover@firewall.loc

sysadmin@firewall.loc

}

notification\_email\_from Alexandre.Cassen@firewall.loc

smtp\_server 127.0.0.1

smtp\_connect\_timeout 30

router\_id NGINX\_BACKUP

}

vrrp\_script check\_nginx {

script "/etc/keepalived/check\_nginx.sh"

}

vrrp\_instance VI\_1 {

state BACKUP

interface ens33

virtual\_router\_id 51 # VRRP 路由 ID实例，每个实例是唯一的

priority 90

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

192.168.1.199/24

}

track\_script {

check\_nginx

}

}

[root@xianchaonode2 ~]# cat /etc/keepalived/check\_nginx.sh

#!/bin/bash

#1、判断Nginx是否存活

counter=$(ps -ef |grep nginx | grep sbin | egrep -cv "grep|$$" )

if [ $counter -eq 0 ]; then

    #2、如果不存活则尝试启动Nginx

    service nginx start

    sleep 2

    #3、等待2秒后再次获取一次Nginx状态

    counter=$(ps -ef |grep nginx | grep sbin | egrep -cv "grep|$$" )

    #4、再次进行判断，如Nginx还不存活则停止Keepalived，让地址进行漂移

    if [ $counter -eq 0 ]; then

        service  keepalived stop

    fi

fi

[root@xianchaonode2 ~]# chmod +x /etc/keepalived/check\_nginx.sh

#注：keepalived根据脚本返回状态码（0为工作正常，非0不正常）判断是否故障转移。

4、启动服务：

[root@xianchaonode1 ~]# systemctl daemon-reload

[root@xianchaonode1 ~]# systemctl enable nginx keepalived

[root@xianchaonode1 ~]# systemctl start nginx

[root@xianchaonode1 ~]# systemctl start keepalived

4、启动服务：

[root@xianchaonode2 ~]# systemctl daemon-reload

[root@xianchaonode2 ~]# systemctl enable nginx keepalived

[root@xianchaonode2 ~]# systemctl start nginx

[root@xianchaonode2 ~]# systemctl start keepalived

5、测试vip是否绑定成功

[root@xianchaonode1 ~]# ip addr

1: lo: <LOOPBACK,UP,LOWER\_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000

link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00

inet 127.0.0.1/8 scope host lo

valid\_lft forever preferred\_lft forever

inet6 ::1/128 scope host

valid\_lft forever preferred\_lft forever

2: ens33: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc pfifo\_fast state UP group default qlen 1000

link/ether 00:0c:29:79:9e:36 brd ff:ff:ff:ff:ff:ff

inet 192.168.40.182/24 brd 192.168.40.255 scope global noprefixroute ens33

valid\_lft forever preferred\_lft forever

inet 192.168.40.199/24 scope global secondary ens33

valid\_lft forever preferred\_lft forever

inet6 fe80::b6ef:8646:1cfc:3e0c/64 scope link noprefixroute

valid\_lft forever preferred\_lft forever

6、测试keepalived：

停掉xianchaonode1上的keepalived。Vip会漂移到xianchaonode2

[root@xianchaonode1 ~]# service keepalived stop

[root@xianchaonode2 ~]# ip addr

1: lo: <LOOPBACK,UP,LOWER\_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000

link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00

inet 127.0.0.1/8 scope host lo

valid\_lft forever preferred\_lft forever

inet6 ::1/128 scope host

valid\_lft forever preferred\_lft forever

2: ens33: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc pfifo\_fast state UP group default qlen 1000

link/ether 00:0c:29:83:4d:9e brd ff:ff:ff:ff:ff:ff

inet 192.168.40.181/24 brd 192.168.40.255 scope global noprefixroute ens33

valid\_lft forever preferred\_lft forever

inet 192.168.40.199/24 scope global secondary ens33

valid\_lft forever preferred\_lft forever

inet6 fe80::a5e0:c74e:d0f3:f5f2/64 scope link tentative noprefixroute dadfailed

valid\_lft forever preferred\_lft forever

inet6 fe80::b6ef:8646:1cfc:3e0c/64 scope link noprefixroute

valid\_lft forever preferred\_lft forever

inet6 fe80::91f:d383:3ce5:b3bf/64 scope link tentative noprefixroute dadfailed

valid\_lft forever preferred\_lft forever

启动xianchaonode1上的keepalived。Vip又会漂移到xianchaomaster1

[root@xianchaonode1 ~]# service keepalived start

[root@xianchaonode1 ~]# ip addr

1: lo: <LOOPBACK,UP,LOWER\_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000

link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00

inet 127.0.0.1/8 scope host lo

valid\_lft forever preferred\_lft forever

inet6 ::1/128 scope host

valid\_lft forever preferred\_lft forever

2: ens33: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc pfifo\_fast state UP group default qlen 1000

link/ether 00:0c:29:79:9e:36 brd ff:ff:ff:ff:ff:ff

inet 192.168.40.182/24 brd 192.168.40.255 scope global noprefixroute ens33

valid\_lft forever preferred\_lft forever

inet 192.168.40.199/24 scope global secondary ens33

valid\_lft forever preferred\_lft forever

inet6 fe80::b6ef:8646:1cfc:3e0c/64 scope link noprefixroute

valid\_lft forever preferred\_lft forever

### 10.3.2 测试Ingress HTTP代理k8s内部pod

1.部署后端tomcat服务

[root@xianchaomaster1 ~]# cat ingress-demo.yaml

apiVersion: v1

kind: Service

metadata:

name: tomcat

namespace: default

spec:

selector:

app: tomcat

release: canary

ports:

- name: http

targetPort: 8080

port: 8080

- name: ajp

targetPort: 8009

port: 8009

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: tomcat-deploy

namespace: default

spec:

replicas: 2

selector:

matchLabels:

app: tomcat

release: canary

template:

metadata:

labels:

app: tomcat

release: canary

spec:

containers:

- name: tomcat

image: tomcat:8.5.34-jre8-alpine

imagePullPolicy: IfNotPresent

ports:

- name: http

containerPort: 8080

name: ajp

containerPort: 8009

#更新资源清单yaml文件：

[root@xianchaomaster1 ~]# kubectl apply -f ingress-demo.yaml

service/tomcat created

deployment.apps/tomcat-deploy created

#查看pod是否部署成功

[root@xianchaomaster1 ~]# kubectl get pods -l app=tomcat

NAME READY STATUS RESTARTS AGE

tomcat-deploy-66b67fcf7b-9h9qp 1/1 Running 0 32s

tomcat-deploy-66b67fcf7b-hxtkm 1/1 Running 0 32s

2、编写ingress规则

#编写ingress的配置清单

[root@xianchaomaster1 ~]# cat ingress-myapp.yaml

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: ingress-myapp

namespace: default

spec:

ingressClassName: nginx

rules:

- host: tomcat.lucky.com

http:

paths:

- backend:

service:

name: tomcat

port:

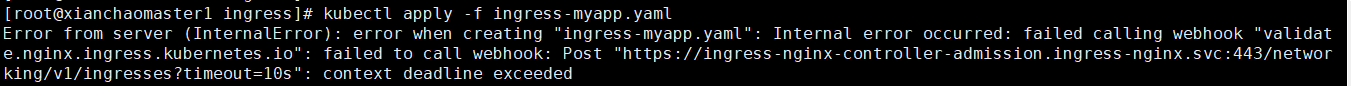
number: 8080

path: /

pathType: Prefix

[root@xianchaomaster1t]# kubectl apply -f ingress-myapp.yaml

报错如下：



解决办法：

[root@xianchaomaster1 ]# kubectl delete -A ValidatingWebhookConfiguration ingress-nginx-admission

[root@xianchaomaster1 ]# kubectl apply -f ingress-myapp.yaml

#查看ingress-myapp的详细信息

[root@xianchaomaster1 ~]# kubectl describe ingress ingress-myapp

Name: ingress-myapp

Namespace: default

Address:

Default backend: default-http-backend:80 (10.244.187.118:8080)

Rules:

Host Path Backends

---- ---- --------

tomcat.lucky.com

tomcat:8080 (10.244.209.172:8080,10.244.209.173:8080)

Annotations: kubernetes.io/ingress.class: nginx

Events:

Type Reason Age From Message

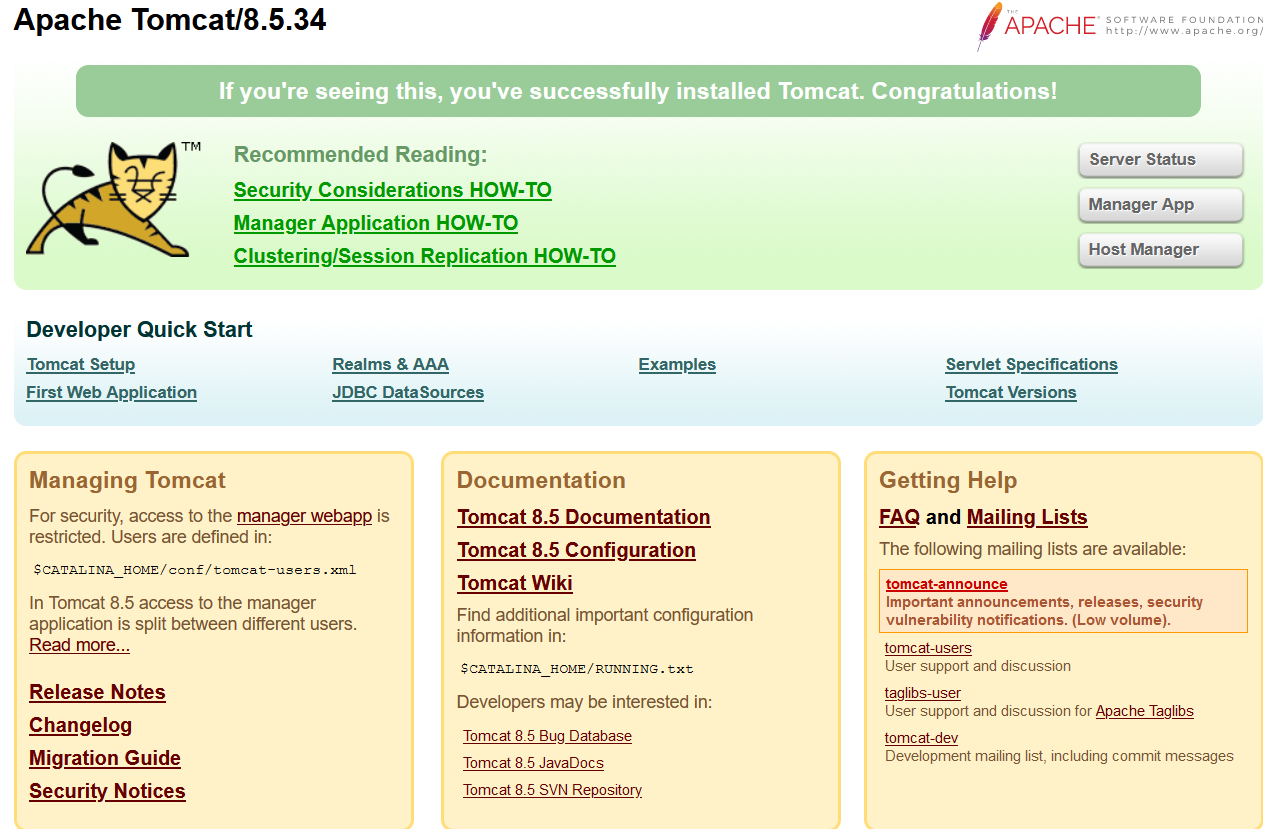
---- ------ ---- ---- -------

Normal CREATE 22s nginx-ingress-controller Ingress default/ingress-myapp

#修改电脑本地的host文件，增加如下一行，下面的ip是xianchaonode1节点ip

192.168.1.199 tomcat.lucky.com

浏览器访问tomcat.lucky.com，出现如下页面：



代理流程：

tomcat.lucky.com:30080->192.168.40.199:30080->192.168.40.181:80,192.168.40.182:80->svc: tomcat:8080->tomcat-deploy-

### 扩展：测试Ingress HTTPS代理k8s内部pod

1、构建TLS站点

（1）准备证书，在xianchaomaster1节点操作

[root@xianchaomaster1]# cd /root/

[root@xianchaomaster1 ~]# openssl genrsa -out tls.key 2048

[root@xianchaomaster1 ~]# openssl req -new -x509 -key tls.key -out tls.crt -subj /C=CN/ST=Beijing/L=Beijing/O=DevOps/CN=hanxianchao.lucky.com

（2）生成secret，在xianchaomaster1节点操作

[root@xianchaomaster1 ~]# kubectl create secret tls tomcat-ingress-secret --cert=tls.crt --key=tls.key

（3）查看secret

[root@xianchaomaster1 ~]# kubectl get secret

显示如下：

tomcat-ingress-secret   kubernetes.io/tls                     2      56s

（4）查看tomcat-ingress-secret详细信息

[root@xianchaomaster1 ~]# kubectl describe secret tomcat-ingress-secret

显示如下：

Name: tomcat-ingress-secret

Namespace: default

Labels: <none>

Annotations: <none>

Type: kubernetes.io/tls

Data

====

tls.crt: 1294 bytes

tls.key: 1679 bytes

2、创建Ingress

Ingress规则可以参考官方：

<https://kubernetes.io/zh/docs/concepts/services-networking/ingress/>

[root@xianchaomaster1 ]# kubectl explain ingress.spec.rules.http.paths.backend.service

[root@xianchaomaster1 ~]# cat ingress-tomcat-tls.yaml

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: ingress-tomcat-tls

namespace: default

spec:

ingressClassName: nginx

tls:

- hosts:

- hanxianchao.lucky.com

secretName: tomcat-ingress-secret

rules:

- host: hanxianchao.lucky.com

http:

paths:

- path: /

pathType: Prefix

backend:

service:

name: tomcat

port:

number: 8080

更新yaml文件

[root@xianchaomaster1]# kubectl delete -f ingress-myapp.yaml

[root@xianchaomaster1 ~]# kubectl apply -f ingress-tomcat-tls.yaml

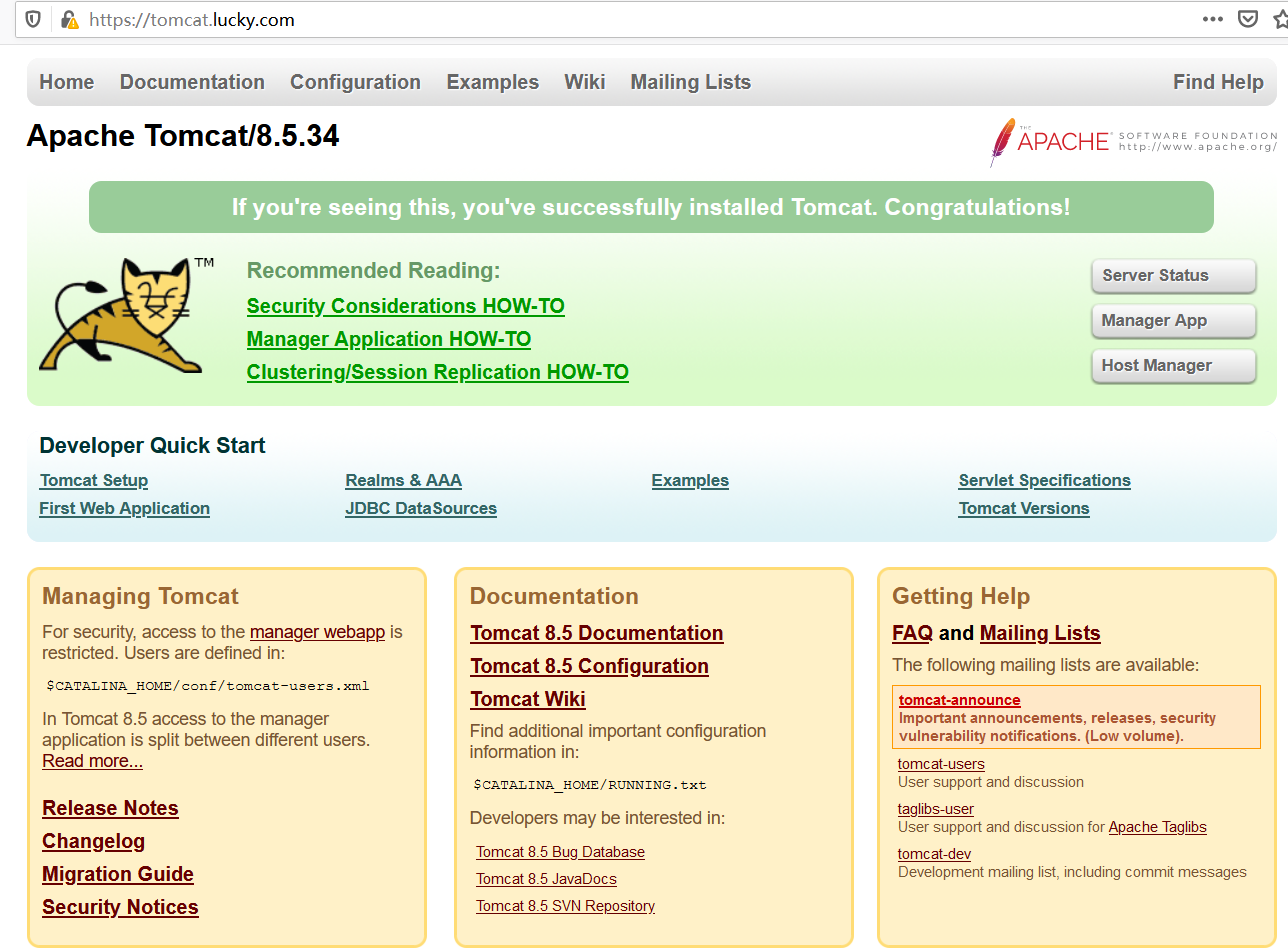
修改电脑的hosts文件：

打开C:\Windows\System32\drivers\etc目录，找到hosts文件，如下：

192.168.40.199 tomcat.lucky.com

192.168.40.199 hanxianchao.lucky.com

浏览器访问https://hanxianchao.lucky.com:30080



### 10.3.3 同一个k8s搭建多套Ingress-controller

ingress可以简单理解为service的service，他通过独立的ingress对象来制定请求转发的规则，把请求路由到一个或多个service中。这样就把服务与请求规则解耦了，可以从业务维度统一考虑业务的暴露，而不用为每个service单独考虑。

在同一个k8s集群里，部署两个ingress nginx。一个deploy部署给A的API网关项目用。另一个daemonset部署给其它项目作域名访问用。这两个项目的更新频率和用法不一致，暂时不用合成一个。

为了满足多租户场景，需要在k8s集群部署多个ingress-controller，给不同用户不同环境使用。

主要参数设置：  
containers:

- name: nginx-ingress-controller

image: registry.cn-hangzhou.aliyuncs.com/google\_containers/nginx-ingress-controller:v1.1.0

args:

- /nginx-ingress-controller

- --ingress-class=ngx-ds

注意：--ingress-class设置该Ingress Controller可监听的目标Ingress Class标识；注意：同一个集群中不同套Ingress Controller监听的Ingress Class标识必须唯一，且不能设置为nginx关键字（其是集群默认Ingress Controller的监听标识）；

创建Ingress规则：

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: ingress-myapp

namespace: default

spec:

ingressClassName: nginx

rules:

- host: tomcat.lucky.com

http:

paths:

- path: /

pathType: Prefix

backend:

service:

name: tomcat

port:

number: 8080

ingressClassName: nginx-ds

# 注意：这里要设置为你前面配置的`controller.ingressClass`唯一标识