

# 关于本文档

---

文档名称：《Kubernetes 流量调度 Ingress》

使用协议：《知识共享公共许可协议(CCPL)》

## 贡献者

---

贡献者名称	贡献度	文档变更记录	个人主页
马哥（马永亮）	主编		<a href="http://github.com/iKubernetes/">http://github.com/iKubernetes/</a>
王晓春	作者	kubernetes-v1.32.0	<a href="http://www.wangxiaochun.com">http://www.wangxiaochun.com</a>

## 文档协议

---

**署名要求：**使用本系列文档，您必须保留本页中的文档来源信息，具体请参考《知识共享 (Creative Commons) 署名4.0公共许可协议国际版》。

**非商业化使用：**遵循《知识共享公共许可协议(CCPL)》，并且您不能将本文档用于马哥教育相关业务之外的其他任何商业用途。

**您的权利：**遵循本协议后，在马哥教育相关业务之外的领域，您将有以下使用权限：

共享 — 允许以非商业性质复制本作品。

改编 — 在原基础上修改、转换或以本作品为基础进行重新编辑并用于个人非商业使用。

## 致谢

---

本文档中，部分素材参考了相关项目的文档，以及通过搜索引擎获得的内容，这里先一并向相关的贡献者表示感谢。

# Kubernetes 流量调度 Ingress

---

## 本章内容

---

- Ingress 原理
- Ingress-nginx 安装和配置
- Ingress-nginx 实现

## • Ingress-nginx 实现蓝绿和灰度发布

# 1 Ingress 原理

```
https://kubernetes.github.io/ingress-nginx/  
https://kubernetes.io/zh-cn/docs/concepts/services-networking/ingress/  
https://kubernetes.io/zh-cn/docs/concepts/services-networking/ingress-controllers/
```

## 1.1 Ingress 工作原理

Kubernetes集群中的通信的不仅有内部的流量，还有外部流量

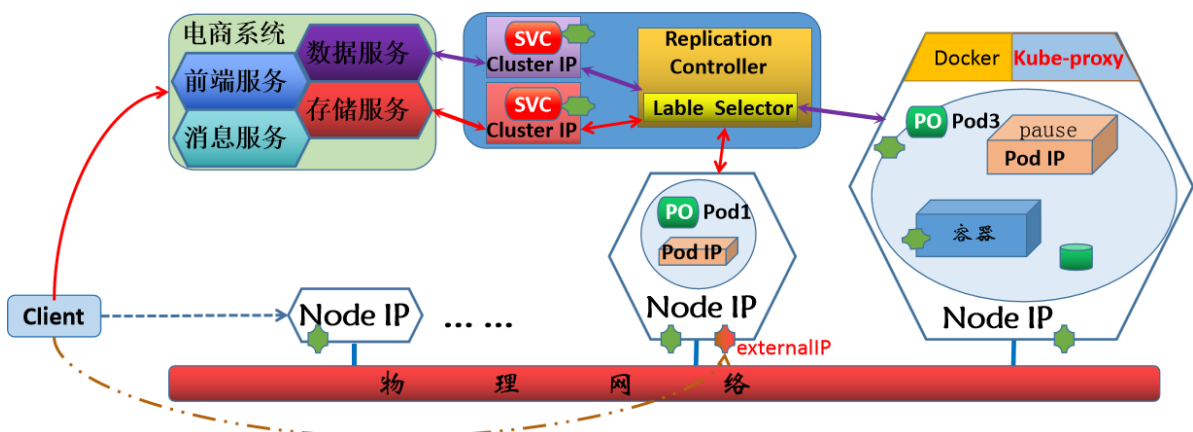
流量方向

- 将集群内部 pod之类对象之间的网络通信称为东西向流量
- 将集群外部应用和集群内部之间对象 之间的网络通信称为南北向流量
  - 南入↓ -- ingress，从集群外到集群内
  - 北出↑ -- egress，从集群内到集群外

可以基于四层的两种常用的方式实现将集群外部的流量引入到集群的内部中来，从而实现外部客户的正常访问。

- service方式：nodePort、LoadBalancer、externalIP 等service对象方式，借助于 namespace、iptables、ipvs 等四层反向代理实现流量的转发
- Host方式：通过node节点的 hostNetwork 与 hostPort 及配套的port映射，以间接的方式实现 service的效果

kubernetes网络

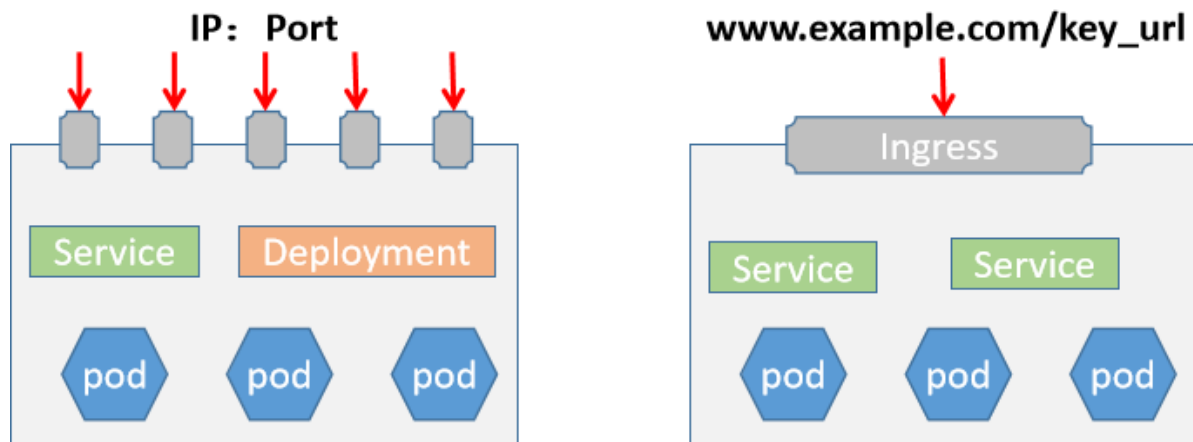


四层方式存在的问题

- 基于service或host等多种方式实现集群内外的网络通信都是基于四层协议来进行调度的，而且应用级别的健康检查功能很难实现。
- 对于外部流量的接入，一般都是基于http(s)协议通信，四层协议无法实现基于应用层的代理,尤其是涉及到各种ssl会话的管理
- 不支持基于FQDN的方式进行访问应用

- 不支持基于URL等机制对HTTP/HTTPS协议进行高级路由、超时/重试、基于流量的灰度等高级流量治理机制
- 难以将多个Service流量统一管理,虽然service能够实现将外部流量引入到集群内部,但是其本质上,是通过网络规则方式来进行转发的,将集群内部的服务暴露到外部。这会造成对外过多的地址和端口暴露

## 解决方案



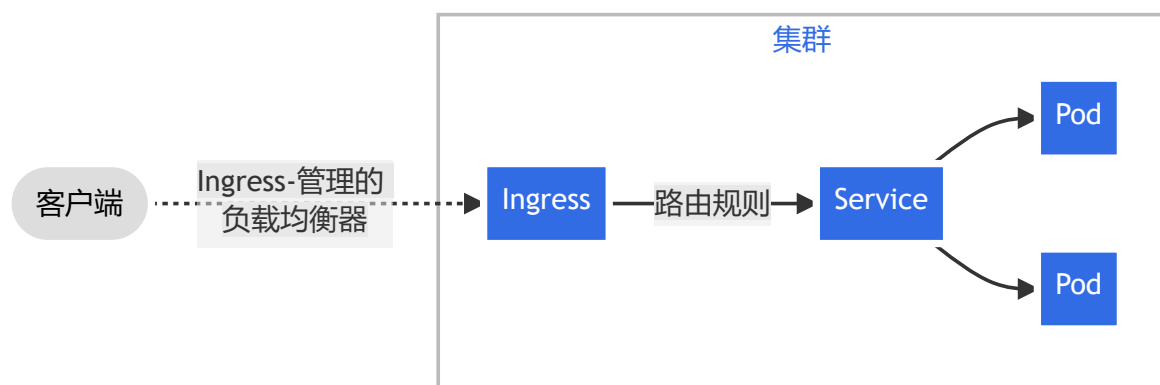
虽然可以自行通过部署七层代理解决上述问题,但配置繁琐以及无法和Kubernetes集群更好的结合联动.

因此Kubernetes 引入了一种新的资源 ingress, 可以解决上面提出问题:

- Ingress本质就是七层代理, 所以可以基于http/https的方式, 将集群外部的流量统一的引入到集群内部
- 通过一个统一的流量入口, 避免将集群内部大量的端口直接暴露给外部

Ingress公开从集群外部到集群内服务 HTTP 和 HTTPS 路由。流量路由由 Ingress 资源上定义的规则控制。

下面是一个将所有流量都发送到同一 Service 的简单 Ingress 示例:



1

Ingress 可为 Service 提供外部可访问的 URL、负载均衡流量、终止 SSL/TLS, 以及基于名称的虚拟托管。 [Ingress 控制器](#) 通常负责通过负载均衡器来实现 Ingress, 尽管它也可以配置边缘路由器或其他前端来帮助处理流量。

Ingress 不会公开任意端口或协议。将 HTTP 和 HTTPS 以外的服务公开到 Internet 时, 通常使用 [Service.Type=NodePort](#) 或 [Service.Type=LoadBalancer](#) 类型的 Service。

Ingress这种利用应用层协议来进行流量的负载均衡效果, 它可以实现让用户通过域名来访问相应的 service就可以了, 无需关心Node IP及Port是什么, 避免了信息的泄露。

Ingress 是kubernetes上的一种标准化资源格式，它为了剥离与特定负载均衡功能实现软件的相关性，而抽象出来的一种公共的统一的声明式配置格式。然后借助于特定的ingress controller功能负责将其加载并转换成k8s集群内部的配置信息。

Ingress是授权入站连接到集群服务的规则集合。

ingress 主要包含两个组件Ingress API和Ingress Controller

ingress 其具备了动态更新并加载新配置的特性。而且ingress本身是不具备实现集群内外流量通信的功能的，这个功能是通过 controller来实现的。Ingress Controller本身是运行于集群中的Pod资源对象

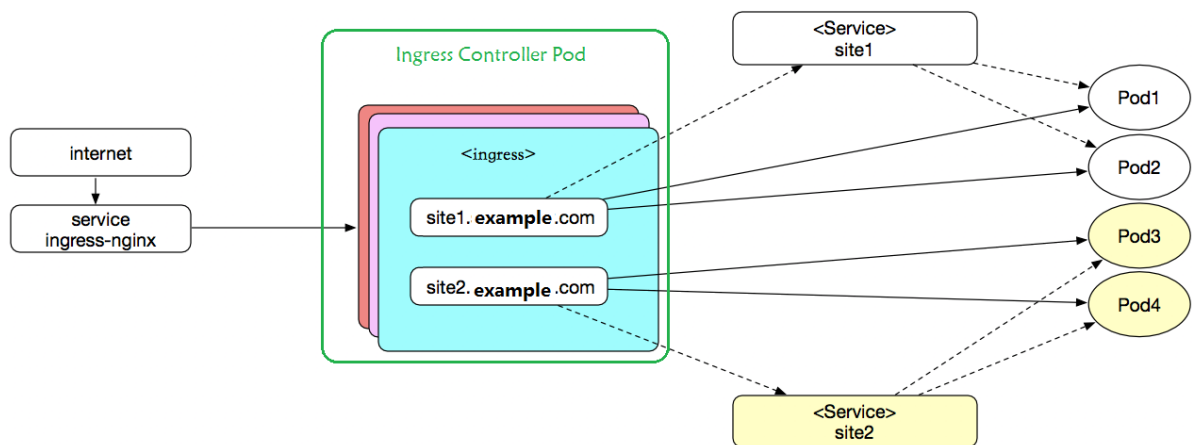
组件	解析
Ingress API	Kubernetes上的标准API资源类型之一 仅定义了抽象路由配置信息，只是元数据，需要由相应的控制器动态加载 将代理配置抽象成一个Ingress对象，每个服务对应一个yaml配置文件 负责以k8s标准的资源格式定义流量调度、路由等规则 属于名称空间级资源,完成将同一个名空间的service资源进行暴露
Ingress Controller	七层反向代理服务程序 需要监视（watch）API Server上 Ingress资源的变动，并生成具体应用的自身的配置文件格式，即将新加入的Ingress转化成反向代理的配置文件并动态加载使之生效，最终并据此完成流量转发 Ingress Controller非为内置的控制器，需要额外自行部署 通常以Pod形式运行于Kubernetes集群之上 一般应该由专用的LB Service负责为其接入集群外部流量

因为ingress Controller是以pod的方式部署的,所以需要解决如下问题

- ingress的pod如何引入外部流量  
通过一个专用的service 即可实现
- 如何实现ingress的Pod的流量负载均衡  
关于pod负载均衡的流量，直接通过deployment/daemonset等controller转发给后端pod即可。
- 后端应用的 Pod 很多，如何找到要转发的目标？  
通过k8s的service对所有的pod进行分组管理，再用controller内部的负载均衡配置，找到对应的目标。  
即后端应用的Pod对应的service 只是起到服务发现Pod的功能，而从外部访问应用的Pod的流量转发过程中不需要再经过此service

### Ingress 访问过程

- 从外部流量调度到kubernetes中Ingress service，有多种实现方案，比如使用节点网络中的EXTERNAL-IP或者NodePort方式
- 从service调度到ingress-controller
- ingress-controller根据ingress Pod 中的定义，比如虚拟主机或者后端的url
- 根据虚拟主机名直接调度到后端的一组应用pod中



注意：

- 整个流程中涉及到了两处service内容
- service ingress-nginx 是帮助 ingress controller Pod 接入外部流量的
- 后端的服务对应的service 只起到帮助 ingress controller Pod 找到具体的服务的Pod，即只用于服务发现，而流量不需要经过后端服务的Service，直接从ingress controller Pod转到至具体的Pod
- 虚线表示service对后端的应用进行分组，实线表示ingress实际的访问流向

## 1.2 Ingress controller 常见的解决方案

对于Ingress controller的软件实现，其实没有特殊的要求，只要能够实现七层的负载均衡功能效果即可

很多组织都提供了Ingress controller的软件实现, 对于Ingress的controller虽然实现方式不同，但是功能基本上一样

Ingress controller 支持由任何具有反向代理功能的程序实现，如Nginx、Traefik、Envoy、HAProxy、Vulcan等

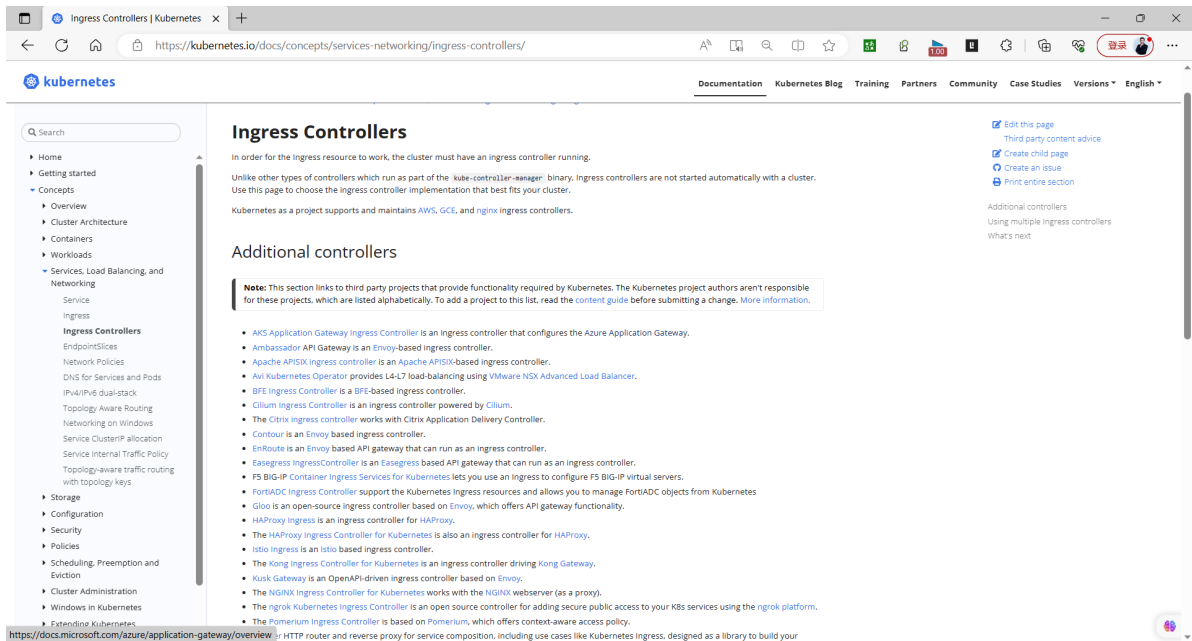
Kubernetes支持同时部署二个或以上的数量的Ingress Controller

Ingress资源配置指定Ingress Controller类型的方法

- 专用的annotation：kubernetes.io/ingress.class，老版本用法,v1.32.0淘汰此方式
- Ingress资源的spec的专有字段：ingressClassName，引用的IngressClass是一种特定的资源类型，此方式v1.18版本起使用，新版本推荐

Ingress controller常见的实现方式有：

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



- Kubernetes Ingress

Kubernetes官方发布, Ingress-Nginx 自从k8s-v1.6版本引入, 在k8s-v1.19 正式进入了一个稳定版本状态。

<http://github.com/kubernetes/ingress-nginx>

- Nginx Ingress

[github.com/nginxinc/kubernetes-ingress](https://github.com/nginxinc/kubernetes-ingress)  
<https://www.f5.com/products/nginx/nginx-ingress-controller>

Nginx Ingress是NGINX开发的官方版本, 它基于NGINX Plus商业版本, NGINX控制器具有很高的稳定性, 持续的向后兼容性, 没有任何第三方模块, 并且由于消除了Lua代码而保证了较高的速度(与官方控制器相比)。

相比官方控制器, 它支持TCP/UDP的流量转发, 付费版有很广泛的附加功能, 主要缺点就是缺失了鉴权方式、流量调度等其他功能。

- Kong Ingress

[github.com/Kong/kubernetes-ingress-controller](https://github.com/Kong/kubernetes-ingress-controller)

Kong Ingress建立在NGINX之上, 并增加了扩展其功能的Lua模块。

kong在之前是专注于API网关, 现在已经成为了成熟的Ingress控制器, 相较于官方控制器, 在路由匹配规则、upstream探针、鉴权上做了提升, 并且支持大量的模块插件, 并且便与配置。

它提供了一些 API、服务的定义, 可以抽象成 Kubernetes 的 CRD, 通过Kubernetes Ingress 配置便可完成同步状态至 Kong 集群。

- HAProxy Ingress

性能相对于nginx来说, 较稳定

<http://github.com/jcmoraisjr/haproxy-ingress>

- Envoy:

基于C++语言开发, 本身就为动态环境设计的, 支持配置动态加载的XDS API

不能像ingress-nginx一样，直接加载配置文件并动态加载转换成对应的应用配置，所以需要借助于其他控制平面工具来实现。

常见解决方案：Contour, Gloo

- Traefik Ingress

Traefik的设计目标是成为一个现代化的云原生应用反向代理和负载均衡器。它专注于与容器化环境（如Docker、Kubernetes）集成，并具有自动发现和动态配置的能力。相比之下，Nginx是一个多用途的Web服务器和反向代理，旨在提供高性能和广泛的功能。

配置语言和方式：Traefik使用声明性的配置语言，通常基于YAML或TOML格式。它支持自动发现和动态配置，可以根据容器的状态和标签进行自动路由。Nginx使用基于文本文件的配置语言，通常使用Nginx专有的配置语法。

由于Traefik专注于容器环境，并提供自动配置功能，对于那些熟悉Docker和Kubernetes的用户来说，使用Traefik可能更加容易上手

当前使用得较多的是 traefik 和 nginx-controller，traefik 的性能较 nginx-controller 差，但是配置使用要简单许多

```
https://traefik.io/
https://github.com/containous/traefik
https://github.com/traefik/traefik/releases
https://github.com/traefik/traefik/tree/v1.7/examples/k8s
```

各种 Ingress Controller 比较

[https://docs.google.com/spreadsheets/d/1DnsHtdHbxjvHmxvlu7VhzWcwGLAN\\_Mc5L1WlhLDA\\_\\_k/edit?spm=a2c6h.12873639.article-detail.4.1eff3d8d053Ifj#gid=0](https://docs.google.com/spreadsheets/d/1DnsHtdHbxjvHmxvlu7VhzWcwGLAN_Mc5L1WlhLDA__k/edit?spm=a2c6h.12873639.article-detail.4.1eff3d8d053Ifj#gid=0)

<a href="#">full article</a>	Kubernetes Ingress	NGINX Ingress	Kong Ingress	Traefik	HAproxy	Voyager	Contour	Istio Ingress	Ambassador	Gloo	Skipper
Protocols	http/https, http2, grpc, tcp/udp (partial)	http/https, http2, grpc, tcp/udp	http/https, http2, grpc, tcp (I4)	http/https, http2 (h2c), grpc, tcp, tcp+tls	http/https, http2, grpc, tcp, tcp+tls	http/https, http2, grpc, tcp, tcp+tls	http/https, http2, grpc, tcp+tls	http/https, http2, grpc, tcp+tls, mongo, mysql, redis	http/https, http2, grpc, tcp+tls	http/https, http2, grpc, tcp, tcp+tls	http/https
Based on	nginx	nginx/nginx plus	nginx	traefik	haproxy	haproxy	envoy	envoy	envoy	envoy	—
Traffic routing	host, path (with regex)	host, path, header, method, query param (all with regex expect host)	host, path, method, header*	host (regex), path (regex), headers (regex), query, path prefix, method	host, path	host, path	host, path	host, path, method, header (all with regex)	host, path, method, header (all with regex)	host, path, method, header (all with regex)	host, path, method, header (all with regex)
Namespace limitations	All cluster or specified namespaces	All cluster or specified namespaces	Specified namespace	All cluster or specified namespaces	All cluster or specified namespaces	All cluster or specified namespaces	All cluster or specified namespaces	All cluster or specified namespaces	All cluster or specified namespaces	All cluster or specified namespaces	All cluster or specified namespaces
Traffic distribution	canary, a/b (cookie balancing)	canary, a/b (routing rules), blue-green (service in the upstream)	canary, acl, blue-green, proxy caching*	canary, blue-green, shadowing	blue-green, shadowing	canary, blue-green, acl	canary, blue-green	canary, a/b, shadowing, http headers, acl, whitelist	canary, a/b, shadowing, http headers, acl, whitelist	canary, shadowing	canary, a/b, blue-green, shadowing, whitelist
Upstream probes	retry, timeouts	retry, timeouts, active health checks (based on http probe for pod)*	active, circuit breaker	retry, timeouts, active, circuit breaker	check-uri, check-address, check-port	haproxy healthchecks	timeouts, active	retry, timeouts, active checks, circuit breakers	retry, timeouts, active checks, circuit breakers	retry, timeouts, circuit breakers	retry, timeouts, circuit breaker
Load balancing	round-robin, sticky sessions, least-conn, ip-hash, ewma	round-robin, least-conn, ip-hash, hash, random, least-time*, sticky sessions*	weighted-round-robin, sticky sessions	weighted-round-robin, dynamic-round-robin, sticky sessions	round-robin, static-rr, leastconn, first, source, url, url_param, header, sticky sessions	round-robin, static-rr, leastconn, first, source, url, url_param, header, sticky sessions	round-robin, sticky sessions, weighted-least-request, ring hash, maglev, random, limit conn, limit req	round-robin, sticky sessions, weighted-least-request, ring hash, maglev, random, limit conn, limit req	round-robin, sticky sessions, weighted-least-request, ring hash, maglev, random	round-robin, sticky sessions, least request, random	round-robin, sticky sessions, random
Authentication	Basic, Client cert, external Basic, external OAuth	-	Basic, HMAC, Key, LDAP, OAuth 2.0, PASETO, OpenID Connect**	Basic, auth-url, auth-tls, external auth	Basic, OAuth, Auth TLS	Basic, OAuth, auth-tls, OAuth Google, OAuth GitHub	-	Basic, mutual tls, OpenID, custom auth	Basic, external auth, OAuth, OpenID	Basic*, external auth*, OAuth*, OpenID*, LDAP*	Basic, OAuth, OpenID
Paid subscription	-	+	+	+	+	+	-	-	+	+	-
GUI	-	+ **	+ **	+	-	-	-	-	-	+	-
JWT validation	-	+	+	-	+	-	-	+	+	+	+
Basic DDoS protection	rate limit, limit conn, limit rps, limit rpm, limit-rate-after, limit-whitelist	max-conns, rate limit, rate-limits (with custom annotations)	advanced rate limit*, rate limit, request size limit, request termination, response rate limit	max-conns, rate limit, ip whitelist	limit-rps, limit-connections, limit-whitelist	max-conns, rate limit, whitelist	max-conns, max-request	acl, whitelist, rate limit	rate limit, load shedding	rate limit*	rate limit
Requests tracing	+	+	+	+	+	-	-	+	+	+	+
Config customization	+	+	+	+	+	+	-	+	-	-	+
WAF	lua-resty-waf, ModSecurity	+	Wallarm	-	ModSecurity	-	-	ModSecurity	-	ModSecurity*	-
GitHub: stars	8900	2900	1230	31400	664	1248	2517	24900	3024	2646	2300
commits (contributors)	5574 (582)	871 (57)	791 (71)	3791 (560)	1131 (39)	1323 (64)	2925 (119)	13945 (640)	15069 (162)	1414 (67)	1786 (104)
releases	110	44	27	316	101	86	55	170	547	330	668
	* In paid version only.										
	** Module is available.										

## 2 Ingress-nginx Controller 安装和配置

### 2.1 部署说明



注意: Ingress-nginx 有两个项目,一个Kubernetes维护和Nginx官方维护,下面以Kubernetes官方维护的项目为例

#参考资料:  
<https://github.com/kubernetes/ingress-nginx>  
<https://kubernetes.io/docs/concepts/services-networking/ingress/>








#安装文档:  
<https://kubernetes.github.io/ingress-nginx/deploy/>  
<https://github.com/kubernetes/ingress-nginx/blob/main/docs/deploy/index.md>

版本说明

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

Supported Versions table

Supported versions for the ingress-nginx project mean that we have completed E2E tests, and they are passing for the versions listed. Ingress-Nginx versions **may** work on older versions, but the project does not make that guarantee.

Supported	Ingress-NGINX version	k8s supported version	Alpine Version	Nginx Version	Helm Chart Version
	v1.12.0	1.32, 1.31, 1.30, 1.29, 1.28	3.21.0	1.25.5	4.12.0
	v1.12.0-beta.0	1.32, 1.31, 1.30, 1.29, 1.28	3.20.3	1.25.5	4.12.0-beta.0
	v1.11.4	1.30, 1.29, 1.28, 1.27, 1.26	3.21.0	1.25.5	4.11.4
	v1.11.3	1.30, 1.29, 1.28, 1.27, 1.26	3.20.3	1.25.5	4.11.3
	v1.11.2	1.30, 1.29, 1.28, 1.27, 1.26	3.20.0	1.25.5	4.11.2
	v1.11.1	1.30, 1.29, 1.28, 1.27, 1.26	3.20.0	1.25.5	4.11.1
	v1.11.0	1.30, 1.29, 1.28, 1.27, 1.26	3.20.0	1.25.5	4.11.0
	v1.10.6	1.30, 1.29, 1.28, 1.27, 1.26	3.21.0	1.25.5	4.10.6
	v1.10.5	1.30, 1.29, 1.28, 1.27, 1.26	3.20.3	1.25.5	4.10.5
	v1.10.4	1.30, 1.29, 1.28, 1.27, 1.26	3.20.0	1.25.5	4.10.4
	v1.10.3	1.30, 1.29, 1.28, 1.27, 1.26	3.20.0	1.25.5	4.10.3
	v1.10.2	1.30, 1.29, 1.28, 1.27, 1.26	3.20.0	1.25.5	4.10.2
	v1.10.1	1.30, 1.29, 1.28, 1.27, 1.26	3.19.1	1.25.3	4.10.1
	v1.10.0	1.29, 1.28, 1.27, 1.26	3.19.1	1.25.3	4.10.0
	v1.9.6	1.29, 1.28, 1.27, 1.26, 1.25	3.19.0	1.21.6	4.9.1
	v1.9.5	1.28, 1.27, 1.26, 1.25	3.18.4	1.21.6	4.9.0
	v1.9.4	1.28, 1.27, 1.26, 1.25	3.18.4	1.21.6	4.8.3
	v1.9.3	1.28, 1.27, 1.26, 1.25	3.18.4	1.21.6	4.8.*



## Support Versions table

Ingress-NGINX version	k8s supported version	Alpine Version	Nginx Version
v1.3.1	1.24, 1.23, 1.22, 1.21, 1.20	3.16.2	1.19.10 <sup>+</sup>
v1.3.0	1.24, 1.23, 1.22, 1.21, 1.20	3.16.0	1.19.10 <sup>+</sup>
v1.2.1	1.23, 1.22, 1.21, 1.20, 1.19	3.14.6	1.19.10 <sup>+</sup>
v1.1.3	1.23, 1.22, 1.21, 1.20, 1.19	3.14.4	1.19.10 <sup>+</sup>
v1.1.2	1.23, 1.22, 1.21, 1.20, 1.19	3.14.2	1.19.9 <sup>+</sup>
v1.1.1	1.23, 1.22, 1.21, 1.20, 1.19	3.14.2	1.19.9 <sup>+</sup>
v1.1.0	1.22, 1.21, 1.20, 1.19	3.14.2	1.19.9 <sup>+</sup>
v1.0.5	1.22, 1.21, 1.20, 1.19	3.14.2	1.19.9 <sup>+</sup>
v1.0.4	1.22, 1.21, 1.20, 1.19	3.14.2	1.19.9 <sup>+</sup>
v1.0.3	1.22, 1.21, 1.20, 1.19	3.14.2	1.19.9 <sup>+</sup>
v1.0.2	1.22, 1.21, 1.20, 1.19	3.14.2	1.19.9 <sup>+</sup>
v1.0.1	1.22, 1.21, 1.20, 1.19	3.14.2	1.19.9 <sup>+</sup>
v1.0.0	1.22, 1.21, 1.20, 1.19	3.13.5	1.20.1

<sup>+</sup> This build is *patched against CVE-2021-23017*.

See [this article](#) if you want upgrade to the stable Ingress API.

Installation Guide - NGINX Ingre x +

<https://kubernetes.github.io/ingress-nginx/deploy/>

☰

NGINX Ingress Controller

🔍 Search

kubernetes/ingress-nginx

14.8k Stars · 7.7k Forks

Installation Guide

There are multiple ways to install the NGINX ingress controller:

- with [Helm](#), using the project repository chart;
- with `kubectl apply`, using YAML manifests;
- with specific addons (e.g. for [minikube](#) or [MicroK8s](#)).

On most Kubernetes clusters, the ingress controller will work without requiring any extra configuration. If you want to get started as fast as possible, you can check the [quick start](#) instructions. However, in many environments, you can improve the performance or get better logs by enabling extra features. We recommend that you check the [environment-specific instructions](#) for details about optimizing the ingress controller for your particular environment or cloud provider.

Contents

- [Quick start](#)
- [Environment-specific instructions](#)
- ... [Docker Desktop](#)
- ... [Rancher Desktop](#)
- ... [minikube](#)
- ... [MicroK8s](#)

Table of contents

- Contents
- Quick start
- Pre-flight check
- Local testing
- Online testing
- Environment-specific instructions
- Local development clusters
  - [minikube](#)
  - [MicroK8s](#)
  - [Docker Desktop](#)
  - [Rancher Desktop](#)
- Cloud deployments
  - AWS
    - [Network Load Balancer \(NLB\)](#)
    - [TLS termination in AWS Load Balancer \(NLB\)](#)
    - [NLB Idle Timeouts](#)
  - GCE-GKE
  - Azure
  - Digital Ocean

The screenshot shows the 'Installation Guide' for the Kubernetes ingress-nginx controller. The page is from the GitHub repository 'kubernetes/ingress-nginx' and is titled 'Installation Guide'. It includes a search bar and a table of contents on the right side. The main content area is divided into sections: 'If you have Helm', 'If you don't have Helm', 'Pre-flight check', and 'Attention'. The 'If you have Helm' section provides a command to upgrade the ingress-nginx controller using Helm. The 'If you don't have Helm' section provides a command to apply the manifest using kubectl. The 'Pre-flight check' section mentions that a few pods should start in the 'ingress-nginx' namespace. The 'Attention' section warns that the default manifest may not work on older versions of Kubernetes (1.18 or earlier) due to API deprecations. The table of contents on the right lists various deployment options, including 'Quick start', 'Pre-flight check', 'Local testing', 'Online testing', 'Environment-specific instructions', 'Local development clusters', 'Cloud deployments', 'AWS', 'Network Load Balancer (NLB)', 'TLS termination in AWS Load Balancer (NLB)', 'NLB Idle Timeouts', 'GCE-GKE', 'Azure', 'Digital Ocean', 'Scaleway', 'Exoscale', 'Oracle Cloud Infrastructure', 'OVHcloud', 'Bare metal clusters', 'Miscellaneous', and 'Checking ingress controller version'.

Installation Guide

Search

kubernetes/ingress-nginx  
14.8k Stars · 7.7k Forks

If you have Helm, you can deploy the ingress controller with the following command:

```
helm upgrade --install ingress-nginx ingress-nginx \
--repo https://kubernetes.github.io/ingress-nginx \
--namespace ingress-nginx --create-namespace
```

It will install the controller in the `ingress-nginx` namespace, creating that namespace if it doesn't already exist.

**Info**

This command is *idempotent*:

- if the ingress controller is not installed, it will install it,
- if the ingress controller is already installed, it will upgrade it.

If you don't have Helm or if you prefer to use a YAML manifest, you can run the following command instead:

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.7.0/deploy/static
```

**Info**

The YAML manifest in the command above was generated with `helm template`, so you will end up with almost the same resources as if you had used Helm to install the controller.

**Attention**

If you are running an old version of Kubernetes (1.18 or earlier), please read [this paragraph](#) for specific instructions. Because of api deprecations, the default manifest may not work on your cluster. Specific manifests for supported Kubernetes versions are available within a sub-folder of each provider.

**Pre-flight check**

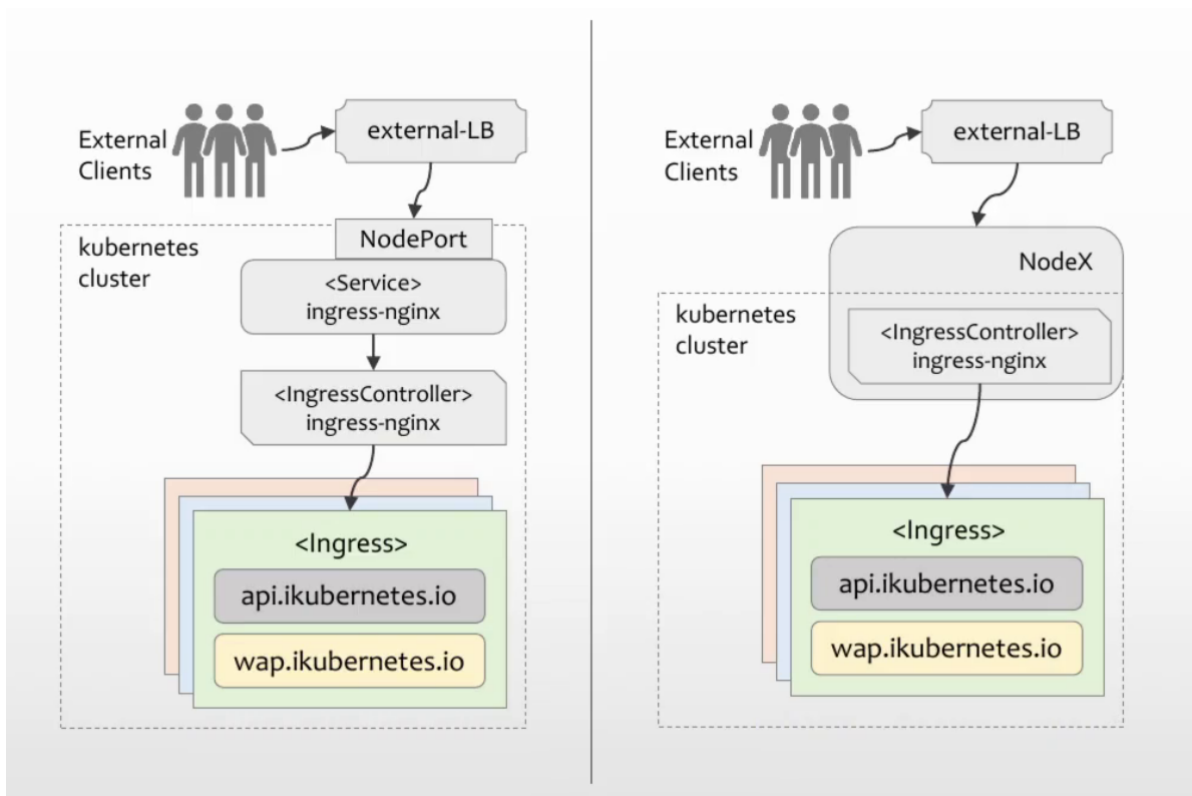
A few pods should start in the `ingress-nginx` namespace:

**Table of contents**

- Contents
- Quick start
- Pre-flight check
- Local testing
- Online testing
- Environment-specific instructions
- Local development clusters
  - minikube
  - MicroK8s
  - Docker Desktop
  - Rancher Desktop
- Cloud deployments
  - AWS
    - Network Load Balancer (NLB)
    - TLS termination in AWS Load Balancer (NLB)
    - NLB Idle Timeouts
  - GCE-GKE
  - Azure
  - Digital Ocean
  - Scaleway
  - Exoscale
  - Oracle Cloud Infrastructure
  - OVHcloud
- Bare metal clusters
- Miscellaneous
  - Checking ingress controller version

## ingress-nginx-controller 部署模式

- 外部用户通过使用ingress-nginx-controller pod所在主机的物理网络访问,可以配置daemonset实现,此方式性能最好
- 外部用户通过ingress-nginx-controller的service对应的外部IP访问
  - 通过ingress-nginx-controller对应service中的externalIPs指定外部IP
  - 使用LB指定ingress-nginx-controller对应NodePort类型Service的节点IP:NodePort 实现
  - 使用ingress-nginx-controller对应LoadBalancer 类型Service的节点IP 实现



## 2.2 部署案例

ingress-nginx 有两种主要的部署方式

- with `kubectl apply`, using YAML manifests
- with Helm, using the project repository chart

### 2.2.1 基于 YAML 部署

范例: 基于 `kubectl apply` 部署

```
#获取配置文件,可能需要科学上网才能下载
https://kubernetes.github.io/ingress-nginx/deploy/

#选择版本
https://github.com/kubernetes/ingress-nginx/

#新版
[root@master1 ~]#VERSION=1.12.0
[root@master1 ~]#VERSION=1.8.2
[root@master1 ~]#VERSION=1.8.0
[root@master1 ~]#VERSION=1.7.0
[root@master1 ~]#VERSION=1.4.0

[root@master1 ~]#wget https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v${VERSION}/deploy/static/provider/cloud/deploy.yaml

[root@master1 ~]#wget https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.8.2/deploy/static/provider/cloud/deploy.yaml

[root@master1 ~]#wget https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.8.0/deploy/static/provider/cloud/deploy.yaml
```

```
[root@master1 ~]#wget https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.7.0/deploy/static/provider/cloud/deploy.yaml
```

#旧版

```
[root@master1 ~]#wget https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.4.0/deploy/static/provider/cloud/deploy.yaml
```

#旧版

```
[root@master1 ~]#wget https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.3.1/deploy/static/provider/cloud/deploy.yaml
```

#旧版

```
[root@master1 ~]#wget https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.0.3/deploy/static/provider/baremetal/deploy.yaml
```

#查看资源

```
[root@master1 ~]#grep '^kind' deploy.yaml
```

```
kind: Namespace
kind: ServiceAccount
kind: ServiceAccount
kind: Role
kind: Role
kind: ClusterRole
kind: ClusterRole
kind: RoleBinding
kind: RoleBinding
kind: ClusterRoleBinding
kind: ClusterRoleBinding
kind: ConfigMap
kind: Service
kind: Service
kind: Deployment
kind: Job
kind: Job
kind: IngressClass
kind: ValidatingWebhookConfiguration
```

#1)默认的镜像可能无法下载,需要修改基础镜像

#可以通过配置代理下载

<https://dockerproxy.com/>

#新版, 修改三处image:地址

```
[root@master1 ~]#vim deploy.yaml
    image: registry.cn-hangzhou.aliyuncs.com/google_containers/nginx-ingress-
controller:v1.12.0
    #image: registry.cn-hangzhou.aliyuncs.com/google_containers/nginx-
ingress-controller:v1.8.2
    #image: k8s.dockerproxy.com/nginx-ingress-controller:v1.8.2
    #image: registry.k8s.io/nginx-
nginx/controller:v1.8.2@sha256:74834d3d25b336b62cabeb8bf7f1d788706e2cf1cfd64022d
e4137ade8881ff2
    #image: k8s.dockerproxy.com/nginx-ingress-kube-webhook-certgen:v20230407
    #image: registry.k8s.io/nginx-ingress-kube-webhook-
certgen:v20230407@sha256:543c40fd093964bc9ab509d3e791f9989963021f1e9e4c9c7b6700b
02bfb227b
    image: registry.cn-hangzhou.aliyuncs.com/google_containers/kube-webhook-
certgen:v1.5.0
```

```
image: registry.cn-hangzhou.aliyuncs.com/google_containers/kube-webhook-  
certgen:v1.5.0
```

```
#image: registry.cn-hangzhou.aliyuncs.com/google_containers/kube-  
webhook-certgen:v20230407
```

#旧版

```
[root@master1 ~]#vim deploy.yaml
```

```
[root@master1 ~]#grep image: deploy.yaml
```

```
image: registry.cn-beijing.aliyuncs.com/wangxiaochun/ingress-nginx-  
controller:v1.8.0
```

```
#image: registry.cn-beijing.aliyuncs.com/wangxiaochun/ingress-nginx-  
controller:v1.7.1
```

```
#image: registry.k8s.io/ingress-  
nginx/controller:v1.7.1@sha256:7244b95ea47bddcb8267c1e625fb163fc183ef55448855e3a  
c52a7b260a60407
```

```
imgae: wangxiaochun/ingress-nginx-kube-webhook-certgen:v20230407
```

```
#image: registry.cn-beijing.aliyuncs.com/wangxiaochun/kube-webhook-  
certgen:v20230312-helm-chart-4.5.2-28-g66a760794
```

```
#image: registry.k8s.io/ingress-nginx/kube-webhook-certgen:v20230312-  
helm-chart-4.5.2-28-  
g66a760794@sha256:01d181618f270f2a96c04006f33b2699ad3ccb02da48d0f89b22abce084b29  
2f
```

```
image: registry.cn-beijing.aliyuncs.com/wangxiaochun/kube-webhook-  
certgen:v20230312-helm-chart-4.5.2-28-g66a760794
```

```
#image: registry.k8s.io/ingress-nginx/kube-webhook-certgen:v20230312-  
helm-chart-4.5.2-28-  
g66a760794@sha256:01d181618f270f2a96c04006f33b2699ad3ccb02da48d0f89b22abce084b29  
2f
```

#旧版

```
[root@master1 ~]#vim deploy.yaml
```

```
[root@master1 ~]#grep image: deploy.yaml
```

```
image: registry.cn-beijing.aliyuncs.com/wangxiaochun/ingress-nginx-  
controller:v1.8.0
```

```
image: registry.cn-beijing.aliyuncs.com/wangxiaochun/ingress-nginx-kube-webhook-  
certgen:v20230407
```

#旧版

```
[root@master1 ~]#vim deploy.yaml
```

```
[root@master1 ~]#grep image: deploy.yaml
```

```
image: registry.cn-beijing.aliyuncs.com/wangxiaochun/ingress-nginx-  
controller:v1.7.0
```

```
image: registry.cn-beijing.aliyuncs.com/wangxiaochun/kube-webhook-  
certgen:v20230312-helm-chart-4.5.2-28-g66a760794
```

#旧版

```
[root@master1 ~]#vim deploy.yaml
```

```
[root@master1 ~]#grep image: deploy.yaml
```

```
image: registry.cn-hangzhou.aliyuncs.com/google_containers/nginx-ingress-  
controller:v1.4.0
```

```
image: registry.cn-hangzhou.aliyuncs.com/google_containers/kube-webhook-  
certgen:v20220916-gd32f8c343
```

#旧版

```
[root@master1 ~]#vim deploy.yaml
```

```
[root@master1 ~]#grep image: deploy.yaml
```

```

        image: registry.cn-hangzhou.aliyuncs.com/google_containers/nginx-
ingress-
controller:v1.0.3@sha256:4ade87838eb8256b094fbb5272d7dda9b6c7fa8b759e6af5383c130
0996a7452
        image: registry.cn-hangzhou.aliyuncs.com/google_containers/kube-
webhook-
certgen:v1.0@sha256:f3b6b39a6062328c095337b4cadcefd1612348fdd5190b1dcbcb9b9e90bd
8068
        image: registry.cn-hangzhou.aliyuncs.com/google_containers/kube-
webhook-
certgen:v1.0@sha256:f3b6b39a6062328c095337b4cadcefd1612348fdd5190b1dcbcb9b9e90bd
8068

```

## #2) 开放外部访问入口地址

```
[root@master1 ~]#vim deploy.yaml
```

```

.....
apiVersion: v1
kind: Service
metadata:
  .....
  name: ingress-nginx-controller
  namespace: ingress-nginx
  annotations:                                #添加如下三行,用于支持prometheus监控,可选
    prometheus.io/scrape: "true"
    prometheus.io/port: "10254"
spec:
  externalTrafficPolicy: Local #默认为Local,只能本机处理流量,即下面externalIPs所在主机
和Ingress-Controller Pod运行的主机必须是同一个节点,如果修改为Cluster,所有节点:3XXX端口
都能处理流量实现负载均衡,如果type是LoadBalancer类型并且已配置openlb类似相关服务,则可以配置
为Local,避免跨主机的网络转发,从而有更好的性能,
  .....
  type: LoadBalancer                        #默认值LoadBalancer可不做修改,即使没有配置Openlb也可以不
做修改,也支持其它类型,比如: NodePort, ClusterIP
  externalIPs: ['10.0.0.99'] #如果type是LoadBalancer类型并且配置openlb则此项可选,增加
一个或多个外网访问的入口IP,如果externalTrafficPolicy为Local则此IP必须是正在运行Ingress-
Controller Pod的节点上面的可用IP,只有此IP能才处理流量,如果为Cluster则可以是任意集群节点的
IP,此项和type无关
  ports:
    - name: http
      port: 80
  .....

```

## #3) 默认ingress-nginx-controller只有一个Pod副本的,可选

#方法1: 指定2个副本实现高可用 (此步可选)

```
spec:
  replicas: 2
```

#方法2可以修改为DaemonSet提高性能 (此步可选)

```

apiVersion: apps/v1
#kind: Deployment
kind: DaemonSet #修改此行为 DaemonSet
.....
spec:
  hostNetwork: true #使用宿主机的网络,直接使用宿主机的IP:80和宿主机的IP:443
  hostPID: true #使用宿主机的Pid

```

## #应用资源配置文件

```
[root@master1 ~]#kubectl apply -f deploy.yaml
```

```

namespace/ingress-nginx created
serviceaccount/ingress-nginx created
serviceaccount/ingress-nginx-admission created
role.rbac.authorization.k8s.io/ingress-nginx created
role.rbac.authorization.k8s.io/ingress-nginx-admission created
clusterrole.rbac.authorization.k8s.io/ingress-nginx created
clusterrole.rbac.authorization.k8s.io/ingress-nginx-admission created
rolebinding.rbac.authorization.k8s.io/ingress-nginx created
rolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
configmap/ingress-nginx-controller created
service/ingress-nginx-controller created
service/ingress-nginx-controller-admission created
deployment.apps/ingress-nginx-controller created
job.batch/ingress-nginx-admission-create created
job.batch/ingress-nginx-admission-patch created
ingressclass.networking.k8s.io/nginx created
validatingwebhookconfiguration.admissionregistration.k8s.io/ingress-nginx-
admission created

```

#查看ingress-nginx-controller的Pod运行在哪个节点上,注意:externalTrafficPolicy: Cluster时无需此步

```

[root@master1 ~]#kubectl get pod -A -o wide |grep ingress-nginx-controller
ingress-nginx    ingress-nginx-controller-56f8b4f667-4rrm9    1/1    Running
0                73s    10.244.2.20    node2.wang.org    <none>    <none>

```

#在k8s集群中的运行ingress-nginx-controller的Pod节点上添加externalIPs指定的IP地址

#注意:externalTrafficPolicy: Cluster时,10.0.0.99绑定在任意集群节点上都可以

#注意: 此方式添加的IP不稳定, 建议写入配置文件的静态IP, 生产使用高可用的VIP

```

[root@node1 ~]#ip a a 10.0.0.99/24 dev eth0 label eth0:1

```

#确认效果

```

[root@master1 ~]#kubectl get ns
NAME                STATUS    AGE
default             Active    5d19h
ingress-nginx       Active    2m41s
kube-node-lease     Active    5d19h
kube-public         Active    5d19h
kube-system         Active    5d19h
kubernetes-dashboard Active    2d19h

```

```

[root@master1 ~]#kubectl get all -n ingress-nginx

```

NAME	READY	STATUS	RESTARTS
AGE			
pod/ingress-nginx-admission-create--1-c857d	0/1	Completed	0
24m			
pod/ingress-nginx-admission-patch--1-gbxzg	0/1	Completed	1
24m			
pod/ingress-nginx-controller-548b8cb9b8-tsxp1	1/1	Running	0
24m			

NAME	TYPE	CLUSTER-IP
EXTERNAL-IP    PORT(S)	AGE	
service/ingress-nginx-controller	NodePort	10.101.159.100
10.0.0.99    80:32453/TCP,443:32699/TCP	24m	
service/ingress-nginx-controller-admission	ClusterIP	10.105.95.216
443/TCP	24m	<none>



NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/ingress-nginx-controller	1/1	1	1	24m

NAME	DESIRED	CURRENT	READY
replicaset.apps/ingress-nginx-controller-548b8cb9b8	1	1	1
24m			

NAME	COMPLETIONS	DURATION	AGE
job.batch/ingress-nginx-admission-create	1/1	4s	24m
job.batch/ingress-nginx-admission-patch	1/1	5s	24m

#### #查看configmap

```
[root@master1 ~]#kubectl get cm -n ingress-nginx ingress-nginx-controller -o
yaml
apiVersion: v1
data:
  allow-snippet-annotations: "true"
kind: ConfigMap
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","data":{"allow-snippet-
annotations":"true"},"kind":"ConfigMap","metadata":{"annotations":{},"labels":
{"app.kubernetes.io/component":"controller","app.kubernetes.io/instance":"ingres
s-nginx","app.kubernetes.io/name":"ingress-nginx","app.kubernetes.io/part-
of":"ingress-nginx","app.kubernetes.io/version":"1.7.0"},"name":"ingress-nginx-
controller","namespace":"ingress-nginx"}}
  creationTimestamp: "2023-04-02T09:47:23Z"
  labels:
    app.kubernetes.io/component: controller
    app.kubernetes.io/instance: ingress-nginx
    app.kubernetes.io/name: ingress-nginx
    app.kubernetes.io/part-of: ingress-nginx
    app.kubernetes.io/version: 1.7.0
  name: ingress-nginx-controller
  namespace: ingress-nginx
  resourceVersion: "253620"
  uid: 3ae6b24d-a92e-4896-9937-e2bf7c1aebe2
```

#### #查看secret

```
[root@master1 ~]#kubectl get secrets -n ingress-nginx
NAME                TYPE      DATA   AGE
ingress-nginx-admission  Opaque    3       18d
```

```
[root@master1 ~]#kubectl get secrets -n ingress-nginx ingress-nginx-admission -
o yaml
apiVersion: v1
data:
```

```
ca:
LS0tLS1CRudJTiBDRVJUSUZJQ0FURS0tLS0tck1JSUJkakNDQVJlZ0F3SUJBZ0lRRFBWbnFvcWVycU9M
UmhuRXVkdQV3REFLQmdncwVhRak9QUVFEQWpBUE1RMHCKQ3dZRFZRUUtFd1J1YVd3eE1DQVhEVE16TURR
d0lqQTVORE15TkZvWUR6SXhNak13TXpBNUIEazBNakkw2pBUApNUTB3Q3dZRFZRUUtFd1J1YVd3eE1G
a3dFd1lIs29aSXpqMENBUVlJS29aSXpqMERBUWNEUwdBRVlDMTNaSHMwClhyZkRus3hhZk1kb1VRSVpq
SDJHnjYycFNsc0svNHJoYjM2b1N5YmpEeTVMb1J4bnJwQ0xmMUZENXJBclBvZXUKV1JOWjRRdHFuM3F1
WUtoWE1GVxdEZ1lEVlIwUEFRSC9CQVFEQWdJRUICTUdBMVVs1FRtU1Bb0dDQ3NHQVFRgpCd01CTUE4
R0ExVWRfD0VCL3dRRk1BTUJBZjh3SFFZRFZSMc9CQ1lFRkf5WVRkbHJ6K1Vkn1E2NUw1clD5V0lwCjdo
dzJNQW9HQ0NxR1NNNDlCQU1DQTBrcQU1FWUNJUURRZ0p0dDRUa1pndDhyMkw2MFY1Zk1aZXNpeHBFUCTM
bmckL1NBUEFKNnFKd0loQU9qNEM0VUNGaEc2Tz1qVERkcXZ0Rmd1K1VXOWZDNvpwR3RoSFhpL3ArMzgK
LS0tLS1FTkQgQ0VSVE1GSUNBEU0tLS0tLQo=
```

```
cert:
LS0tLS1CRudJTiBDRVJUSUZJQ0FURS0tLS0tck1JSUJlekdQVdHZ0F3SUJBZ0lSQUXok2JYR3hIdlZP
UENma0F6czVkr1l3Q2dzSutvWk16ajBFQXdjd0R6RU4KTUFR0ExVUVDaE1FYm1sc01UQWdGdzB5TXpB
ME1ESXdPVFF5TWpsYUdB0HlNVE16TURnd09UQTVORE15TkZvdWpEekVOTUFzR0ExVUVDaE1FYm1sc01q
Q1pnQk1HQnlxR1NNNDlBZ0VHQ0NxR1NNNDlBd0VIQTBJQUJlIwYXN3C1hlcGRQUFhryKRuUGQ0Z1du
TmErZ0V5OFN0Tj1lXWRCb3lFRExzR2ZNNzVaeGZiVUNaV2QzVGVAaktMYnVjOFAKbXpyUEpsU1VnTERF
WmF5amdac3dnwmd3RGdZRFZSMFBBUgVqFRREFnV2dNQk1HQTFVZEprU1NQW9HQ0NzRwpBUVVGQndN
Qk1Bd0dBMVVsKXxdFQ93UUNNQUF3WxdZRFZSMFJCnd3V29JawFXNW5jbVZ6Y3kxdVoybHVlQzFqCmIy
NTBjbTlZykdwexXRMtiV2x6YzJsdmJvSTBhVzVuY21wemN5MXVAMmx1ZUMxamIyNTBjbTlZykdwexX
RmsKYldsemMybHZiaTVwYm1keVpYTNpMVZvVUVC1NEXUjJZekFLQmdncwVhRak9QUVFEQWd0SUFEQkZB
aUVBeXJndgpowEFIZ3JwM2R1VW1YNmZUYkt00FAVXFTQ2RNZnErOW5qQkhkc1FDSUIzewhYM1d5aHM1
cktrR0orTjVvdThlCi9RYVlyUGR3YXp5R2x0LzZ5NmxCi0tLS0tRU5EIENFU1RJRklDQVRFLS0tLS0K
key:
```

```
LS0tLS1CRudJTiBFQyBQUk1lWQVRFIEtFWS0tLS0tck1IY0NBUEVFSudON0xc293S0Nsb2NDT0ZvW1p1
K1prRUNraTdpZ1gxRHUzVDRzZWdZM0JvQW9HQ0NxR1NNNDkKQXdFSG9VUURRZ0FFYyt2UnF6dGQ2bDA4
OWVSc05NOTNpQmFjMXI2QVRMeEswMzFaadBhaklRTXV3Wjh6dmxuRgo5dFFkbFozZE41bu1vdHU1encr
Yk9zOG1WS1NBc01sbHJBPT0KLS0tLS1FTkQgRUMgUFJvVkFURSBLRVktLS0tLQo=
```

kind: Secret

metadata:

```
creationTimestamp: "2023-04-02T09:47:24Z"
name: ingress-nginx-admission
namespace: ingress-nginx
resourceVersion: "253670"
uid: ce0ecbe6-af86-4397-8217-169db6a1b38e
```

type: Opaque

#查看基于LoadBalancer的service

```
[root@master1 ~]#kubectl get svc -n ingress-nginx
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP
ingress-nginx-controller	LoadBalancer	10.98.94.113	10.0.0.10
ingress-nginx-controller-admission	ClusterIP	10.101.144.209	<none>

```
[root@master1 ~]#kubectl get ep -n ingress-nginx
```

NAME	ENDPOINTS	AGE
ingress-nginx-controller	10.244.1.126:443,10.244.1.126:80	13m
ingress-nginx-controller-admission	10.244.1.126:8443	13m

```
[root@master1 ~]#curl 10.0.0.10
```

```
<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

```
[root@master1 ~]#curl 10.0.0.101:40090
<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

#查看基于NodePort的service

```
[root@master1 ~]#kubectl get svc -n ingress-nginx
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP
ingress-nginx-controller	NodePort	10.101.159.100	10.0.0.99
ingress-nginx-controller-admission	ClusterIP	10.105.95.216	<none>

#注意：新版已经无法看到端口的监听

```
[root@master1 ~]#ss -ntl|grep -E "32453|32699"
```

LISTEN	0	4096	0.0.0.0:32699	0.0.0.0:*
LISTEN	0	4096	0.0.0.0:32453	0.0.0.0:*

#externalTrafficPolicy: local时，测试访问只有运行controller的节点的IP:32453才能访问，如果为Cluster则支持任意节点

```
[root@master1 ~]#curl 10.0.0.101:32453
<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

```
[root@master1 ~]#curl 10.0.0.101:32699
<html>
<head><title>400 The plain HTTP request was sent to HTTPS port</title></head>
<body>
<center><h1>400 Bad Request</h1></center>
<center>The plain HTTP request was sent to HTTPS port</center>
<hr><center>nginx</center>
</body>
</html>
```

#直接获取状态码

```
[root@master1 ~]#curl -I -o /dev/null -s -w %{http_code}"\n" 10.0.0.101:32453
404
```

#查看访问日志

```
[root@master1 ~]#kubectl logs -f -n ingress-nginx ingress-nginx-controller-548b8cb9b8-tsxpl
```

## 2.2.2 基于 Helm 部署

官方链接:

```
https://kubernetes.github.io/ingress-nginx/deploy/#quick-start
```

范例: 通过Helm 安装 ingress-nginx

```
#先安装helm
略

#安装ingress-nginx
[root@master1 ~]#helm repo add ingress-nginx
https://kubernetes.github.io/ingress-nginx
"ingress-nginx" has been added to your repositories

[root@master1 ~]#helm repo list
NAME                URL
ingress-nginx       https://kubernetes.github.io/ingress-nginx

#搜索相关ingress-nginx包
[root@master1 ~]#helm search repo ingress-nginx
NAME                                CHART VERSION   APP VERSION DESCRIPTION
ingress-nginx/ingress-nginx 4.4.0           1.5.1           Ingress controller for
Kubernetes using NGINX a...

#下载包
[root@master1 ~]#helm pull ingress-nginx/ingress-nginx

#解压缩
[root@master1 ~]#tar xf ingress-nginx-4.4.0.tgz
[root@master1 ~]#ls ingress-nginx/
CHANGELOG.md Chart.yaml ci OWNERS README.md README.md.gotmpl templates
values.yaml

[root@master1 ~]#vim ingress-nginx/values.yaml
.....
image:
  ## Keep false as default for now!
  chroot: false
  #registry: registry.k8s.io
  #image: ingress-nginx/controller
  registry: registry.cn-hangzhou.aliyuncs.com #添加下面两行
  image: google_containers/nginx-ingress-controller
.....
#dnsPolicy: ClusterFirst
dnsPolicy: ClusterFirstWithHostNet #修改此行
.....
hostNetwork: true #false修改此行为true
.....
#kind: Deployment
kind: DaemonSet #修改此行
.....
nodeSelector:
  kubernetes.io/os: linux
```

```

    ingress: "true"    #添加此行
    .....
    #type: LoadBalancer
    type: ClusterIP    #修改此行

root@master1 ~]#cd ingress-nginx/
[root@master1 ingress-nginx]#kubectl create ns ingress-nginx

[root@master1 ingress-nginx]#kubectl label node master1.wang.org ingress=true

[root@master1 ingress-nginx]#helm install ingress-nginx -n ingress-nginx .

#或者通过下面实现安装
[root@master1 ~]#helm upgrade --install ingress-nginx ingress-nginx \
  --repo https://kubernetes.github.io/ingress-nginx \
  --namespace ingress-nginx --create-namespace

```

## 3 Ingress 命令式实现

### 3.1 命令式实现说明

```

#创建Ingress的命令:
kubectl create ingress NAME --rule=domain/url=service:port[,tls=secret]]
[options]

#常用选项
--annotation=[]    #注解信息，格式“annotation=value”
--rule=[]           #代理规则，格式“host/path=service:port[,tls=secretname]”，注意:rule
                    #中外部域名要在所有的名称空间唯一
--class=''          #此Ingress适配的Ingress Class Controller
-n 名称空间         #ingress的名称空间，需要和service在同一个名称空间

#基于URI方式代理不同应用的请求时，后端应用的URI若与代理时使用的URI不同，则需要启用URL Rewrite
完成URI的重写
Ingress-Nginx支持使用“annotation nginx.ingress.kubernetes.io/rewrite-target”注解进行
行

```

### 3.2 命令式实现案例

范例：准备环境实现两个service应用 pod-test1和pod-test2

```
#准备后端的应用pod-test v0.1和相应的service
[root@master1 ~]#kubectl create deployment pod-test1 --image=registry.cn-
beijing.aliyuncs.com/wangxiaochun/pod-test:v0.1 --replicas=3
[root@master1 ~]#kubectl create service clusterip pod-test1 --tcp=80:80
```

```
#准备后端的应用pod-test v0.2和相应的service
[root@master1 ~]#kubectl create deployment pod-test2 --image=registry.cn-
beijing.aliyuncs.com/wangxiaochun/pod-test:v0.2 --replicas=3
[root@master1 ~]#kubectl create service clusterip pod-test2 --tcp=80:80
```

```
[root@master1 ~]#kubectl get endpoints
```

NAME	ENDPOINTS	AGE
pod-test1	10.244.1.60:80,10.244.1.64:80,10.244.1.69:80	21h
pod-test2	10.244.1.65:80,10.244.1.66:80,10.244.1.70:80	21h

## 3.2.1 单域名

### 3.2.1.1 实现单域名不支持子URL

范例: 命令式实现单域名不支持子URL,子URL无法访问返回404

```
#路径精确匹配,对于发往www.wang.org的请求,代理至service/pod-test1,其它的URL则无法代理
[root@master1 ~]#kubectl create ingress demo-ingress --rule="www.wang.org/=pod-
test1:80" --class=nginx -o yaml --dry-run=client
```

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  creationTimestamp: null
  name: demo-ingress
spec:
  ingressClassName: nginx
  rules:
  - host: www.wang.org
    http:
      paths:
      - backend:
          service:
            name: pod-test1
            port:
              number: 80
        path: /
        pathType: Exact #表示精确匹配, --rule="www.wang.org/=pod-test1:80",则为
prefix
status:
  loadBalancer: {}
```

#创建

```
[root@master1 ~]#kubectl create ingress demo-ingress --rule="www.wang.org/=pod-
test1:80" --class=nginx
```

#查看

```
[root@master1 ~]#kubectl get ingress demo-ingress -o yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
```

```

creationTimestamp: "2021-04-20T09:19:39Z"
generation: 1
name: demo-ingress
namespace: default
resourceVersion: "671603"
uid: 5223c7de-c032-45b5-b541-2cc8a136ac1f
spec:
  ingressClassName: nginx
  rules:
  - host: www.wang.org
    http:
      paths:
      - backend:
          service:
            name: pod-test1
            port:
              number: 80
        path: /
        pathType: Exact    #类型为Exact
status:
  loadBalancer:
    ingress:
      - ip: 10.0.0.99

```

```
[root@master1 ~]#kubectl get ingress
```

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
demo-ingress	nginx	www.wang.org	10.0.0.99	80	58s

#查看ingress-nginx-controller对应的Pod中Nginx配置文件的变化

```

[root@master1 ~]#kubectl exec -it ingress-nginx-controller-6fc8fc9c5b-f4bc6 -n
ingress-nginx -- grep wang.org /etc/nginx/nginx.conf
## start server www.wang.org
server_name www.wang.org ;
## end server www.wang.org

```

#集群外访问

```

C:\Users\Wang>curl -H"host: www.wang.org" http://10.0.0.99
kubernetes pod-test v0.1!! ClientIP: 10.244.2.31, ServerName: pod-test1-
5f8f7698d8-44vbk, ServerIP: 10.244.3.14!

```

```

C:\Users\Wang>curl -H"host: www.wang.org" http://10.0.0.99
kubernetes pod-test v0.1!! ClientIP: 10.244.2.31, ServerName: pod-test1-
5f8f7698d8-xzvdf, ServerIP: 10.244.2.32!

```

```

C:\Users\Wang>curl -H"host: www.wang.org" http://10.0.0.99
kubernetes pod-test v0.1!! ClientIP: 10.244.2.31, ServerName: pod-test1-
5f8f7698d8-jm4bq, ServerIP: 10.244.1.30!

```

#访问子URL失败，原因是只发布了www.wang.org的根目录，其它URL没有发布

```

C:\Users\Wang>curl -H"host: www.wang.org" http://10.0.0.99/hostname
<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
<hr><center>nginx</center>

```



```
</body>
</html>
```

#清理

```
[root@master1 ~]#kubectl delete ingress demo-ingress
```

### 3.2.1.2 实现单域名支持子URL

范例：命令式实现单域名支持子URL

#添加/\*，支持子URL，如果有URL则转发至Pod对应相同的URL

```
[root@master1 yam1]#kubectl create ingress demo-ingress --
rule="www.wang.org/*=pod-test1:80" --class=nginx -o yaml --dry-run=client
```

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  creationTimestamp: null
  name: demo-ingress
spec:
  ingressClassName: nginx
  rules:
  - host: www.wang.org
    http:
      paths:
      - backend:
          service:
            name: pod-test1
            port:
              number: 80
        path: /
        pathType: Prefix #注意此处
status:
  loadBalancer: {}
```

#创建

```
[root@master1 ~]#kubectl create ingress demo-ingress --rule="www.wang.org/*=pod-
test1:80" --class=nginx
```

```
[root@master1 ~]#kubectl get ingress
```

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
demo-ingress	nginx	www.wang.org	10.0.0.99	80	58s

#查看

```
[root@master1 ~]#kubectl get ingress demo-ingress -o yaml
```

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  creationTimestamp: "2021-05-28T01:36:20Z"
  generation: 1
  name: demo-ingress
  namespace: default
  resourceVersion: "1502354"
  uid: 0938aeda-0ea6-4235-a032-3a6adb0e2174
spec:
  ingressClassName: nginx
  rules:
```

```

- host: www.wang.org
  http:
    paths:
      - backend:
          service:
            name: pod-test1
            port:
              number: 80
          path: /
          pathType: Prefix
status:
  loadBalancer: {}

```

#### #客户端访问测试

```

C:\Users\Wang>curl -H"host: www.wang.org" 10.0.0.99
kubernetes pod-test v0.1!! ClientIP: 10.244.1.182, ServerName: pod-test1-
5f8f7698d8-p2pst, ServerIP: 10.244.3.105!

```

#### #支持子URL

```

C:\Users\Wang>curl -H"host: www.wang.org" 10.0.0.99/hostname
ServerName: pod-test1-5f8f7698d8-p2pst

```

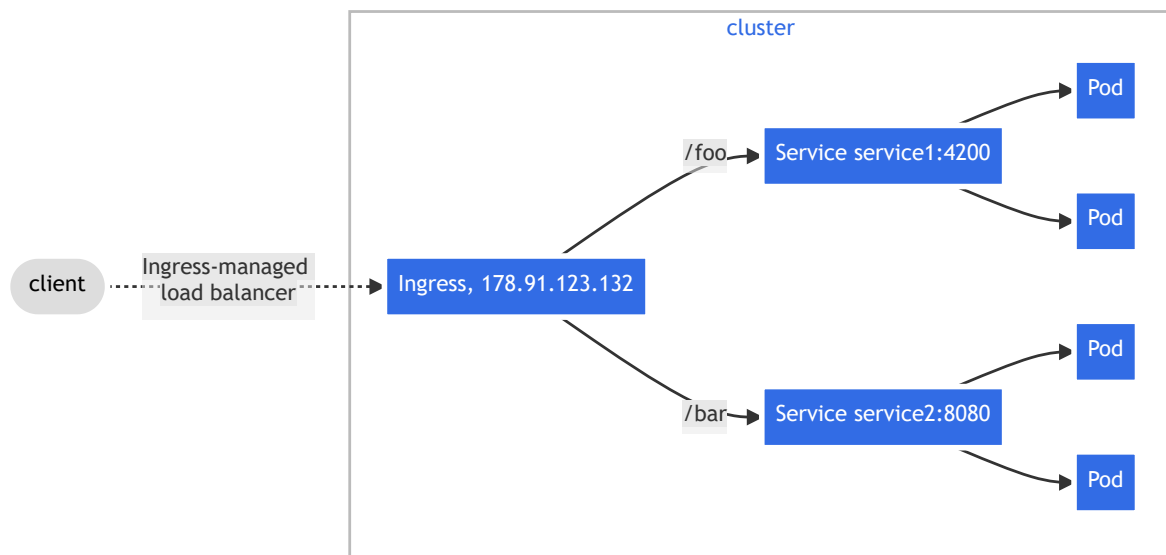
#### #清理

```

[root@master1 ~]#kubectl delete ingress demo-ingress

```

## 3.2.2 单域名多URL



在同一个FQDN下通过不同的URL完成不同应用间的流量分发

### 3.2.2.1 单域名多URL不支持子URL

范例: 命令式实现单域名多URL, 不支持子URL, 如果子URL访问, 也全部转发至后端Pod的根路径 /

#路径精确匹配, 对于发往www.wang.org/v1和www.wang.org/v2的请求, 分别代理至service/pod-test1和service/pod-test2的对应的子URL

```
[root@master1 ~]#kubectl create ingress demo-ingress1 --
rule="www.wang.org/v1=pod-test1:80" --rule="www.wang.org/v2=pod-test2:80" --
class=nginx
```

#集群外访问失败, 原因是后端服务没有对应的/v1这样的子URL资源

```
C:\Users\Wang>curl -H"host: www.wang.org" 10.0.0.99/v1/
<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
<hr><center>nginx</center>
</body>
</html>
C:\Users\Wang>curl -H"host: www.wang.org" 10.0.0.99/v2/
<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

#路径精确匹配, 对于发往www.wang.org/v1和/v2的请求, 分别代理至service/pod-test1和service/pod-test2的根

```
[root@master1 ~]#kubectl create ingress demo-ingress1 --
rule="www.wang.org/v1=pod-test1:80" --rule="www.wang.org/v2=pod-test2:80" --
class=nginx --annotation nginx.ingress.kubernetes.io/rewrite-target="/"
#说明: --annotation nginx.ingress.kubernetes.io/rewrite-target="/"表示代理至后端服务的根/, 而非默认代理至后端服务的子URL/v1和/v2
```

```
[root@master1 ~]#kubectl get ingress
```

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
demo-ingress1	nginx	www.wang.org	10.0.0.99	80	25m

```
[root@master1 ~]#kubectl get ingress demo-ingress1 -o yaml
```

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
  creationTimestamp: "2021-04-20T09:28:50Z"
  generation: 1
  name: demo-ingress1
  namespace: default
  resourceVersion: "672464"
  uid: a4dc438d-d806-4a9b-87cd-cef06f5f2020
spec:
  ingressClassName: nginx
  rules:
  - host: www.wang.org
```

```

http:
  paths:
    - backend:
        service:
          name: pod-test1
          port:
            number: 80
        path: /v1
        pathType: Exact    #类型为Exact
    - backend:
        service:
          name: pod-test2
          port:
            number: 80
        path: /v2
        pathType: Exact
status:
  loadBalancer: {}

[root@master1 ~]#kubectl describe ingress demo-ingress1
Name:          demo-ingress1
Labels:        <none>
Namespace:     default
Address:       10.0.0.99
Ingress Class: nginx
Default backend: <default>
Rules:
  Host          Path    Backends
  ----          -
  www.wang.org   /v1     pod-test1:80 (10.244.1.26:80,10.244.2.5:80,10.244.3.5:80)
                 /v2     pod-test2:80 (10.244.1.27:80,10.244.2.6:80,10.244.3.6:80)
Annotations:    nginx.ingress.kubernetes.io/rewrite-target: /
Events:
  Type    Reason    Age          From          Message
  ----    -
  Normal  Sync      2m56s (x4 over 26m)  nginx-ingress-controller  Scheduled for sync
  Normal  Sync      2m56s (x4 over 26m)  nginx-ingress-controller  Scheduled for sync
  Normal  Sync      2m56s (x4 over 26m)  nginx-ingress-controller  Scheduled for sync
  Normal  Sync      2m56s (x4 over 26m)  nginx-ingress-controller  Scheduled for sync

#查看ingress-nginx controller对应Pod的地址
[root@master1 ~]#kubectl get pod -o wide -n ingress-nginx
NAME                                READY    STATUS    RESTARTS   AGE
IP                                  NODE     NOMINATED NODE   READINESS GATES
ingress-nginx-admission-create-hkg9q 0/1      Completed 0           14m
10.244.2.36 node2.wang.org <none>         <none>
ingress-nginx-admission-patch-9khhf 0/1      Completed 1           14m
10.244.2.35 node2.wang.org <none>         <none>
ingress-nginx-controller-56f8b4f667-kvrff 1/1      Running   0           14m
10.244.2.37 node2.wang.org <none>         <none>

#查看应用的Pod名称
[root@master1 ~]#kubectl get pod -o wide |grep pod-test1

```

```

pod-test1-5f8f7698d8-9vzgm          1/1      Running   2 (10h ago)    18h
10.244.1.60      node1.wang.org   <none>      <none>
pod-test1-5f8f7698d8-jrg5x          1/1      Running   3 (10h ago)    22h
10.244.1.64      node1.wang.org   <none>      <none>
pod-test1-5f8f7698d8-kgv6n          1/1      Running   2 (10h ago)    18h
10.244.1.69      node1.wang.org   <none>      <none>
[root@master1 ~]#

```

#从集群外部测试访问

```

C:\Users\Wang>curl -H"host:www.wang.org" http://10.0.0.99/v1
kubernetes pod-test v0.1!! ClientIP: 10.244.2.26, ServerName: pod-test1-
5f8f7698d8-jrg5x, ServerIP: 10.244.1.64!
C:\Users\Wang>curl -H"host:www.wang.org" http://10.0.0.99/v2
kubernetes pod-test v0.2!! ClientIP: 10.244.2.26, ServerName: pod-test2-
f7d78c544-f415k, ServerIP: 10.244.1.70!

```

#如果有URL，则访问的资源仍然为根目录，不支持对应的子URL

```

C:\Users\Wang>curl -H"host:www.wang.org" http://10.0.0.99/v1/hostname
kubernetes pod-test v0.1!! ClientIP: 10.244.2.26, ServerName: pod-test1-
5f8f7698d8-jrg5x, ServerIP: 10.244.1.64!
C:\Users\Wang>curl -H"host:www.wang.org" http://10.0.0.99/v2/hostname
kubernetes pod-test v0.2!! ClientIP: 10.244.2.26, ServerName: pod-test2-
f7d78c544-f415k, ServerIP: 10.244.1.70!

```

#查看应用的Pod的日志，可以看到客户端来源为ingress-nginx controller对应Pod的地址

```

[root@master1 ~]#kubectl logs pod-test1-5f8f7698d8-9vzgm -f
* Running on http://0.0.0.0:80/ (Press CTRL+C to quit)
10.244.2.34 - - [02/Apr/2021 09:44:42] "GET / HTTP/1.1" 200 -

```

#清理环境

```

[root@master1 ~]#kubectl delete ingress demo-ingress1

```

### 3.2.2.2 单域名多URL支持子URL

范例：命令式实现单域名多URL，支持子URL

#使用URI的前缀匹配，而非精确匹配，且基于正则表达式模式进行url rewrite

#新版变化：kubernetes-1.32.0 使用指令式命令出错，无法实现

```

[root@master1 ~]#kubectl create ingress demo-ingress2 --
rule='www.wang.org/v1(/|$)(.*)=pod-test1:80' --rule='www.wang.org/v2(/|$)
(.*)=pod-test2:80' --class=nginx --annotation
nginx.ingress.kubernetes.io/rewrite-target='/ $2'
warning: path /v1(/|$)(.*) cannot be used with pathType Exact
warning: path /v2(/|$)(.*) cannot be used with pathType Exact
error: failed to create ingress: admission webhook
"validate.nginx.ingress.kubernetes.io" denied the request: ingress contains
invalid paths: path /v1(/|$)(.*) cannot be used with pathType Exact
path /v2(/|$)(.*) cannot be used with pathType Exact

```

#使用清单文件实现

```

[root@master1 ~]#kubectl create ingress demo-ingress2 --
rule='www.wang.org/v1(/|$)(.*)=pod-test1:80' --rule='www.wang.org/v2(/|$)
(.*)=pod-test2:80' --class=nginx --annotation
nginx.ingress.kubernetes.io/rewrite-target='/ $2'> demo-ingress2.yaml

```

```

[root@master1 ~]#vim demo-ingress2.yaml

```

```

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$2
  creationTimestamp: null
  name: demo-ingress2
spec:
  ingressClassName: nginx
  rules:
  - host: www.wang.org
    http:
      paths:
      - backend:
          service:
            name: pod-test1
            port:
              number: 80
          path: /v1(/|$)(.*)
          #pathType: Exact          #默认类型为Exact
          pathType: ImplementationSpecific #修改为类型
      - backend:
          service:
            name: pod-test2
            port:
              number: 80
          path: /v2(/|$)(.*)
          #pathType: Exact          #默认类型为Exact
          pathType: ImplementationSpecific #修改为类型

```

```
[root@master1 ~]#kubectl apply -f demo-ingress2.yaml
```

```

[root@master1 ~]#kubectl create ingress demo-ingress2 --
rule='www.wang.org/v1(/|$)(.*)=pod-test1:80' --rule='www.wang.org/v2(/|$)
(.*)=pod-test2:80' --class=nginx --annotation
nginx.ingress.kubernetes.io/rewrite-target='/$2'

```

#注意: '/\$2'是单引号,不能为双引号

```
[root@master1 ~]#kubectl get ingress
```

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
demo-ingress2	nginx	www.wang.org	10.0.0.99	80	2m41s

```
[root@master1 ~]#kubectl get ingress demo-ingress2 -o yaml
```

```

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$2
  creationTimestamp: "2021-04-02T09:19:56Z"
  generation: 1
  name: demo-ingress2
  namespace: default
  resourceVersion: "251994"
  uid: 1bb2c72d-d834-4bef-a0f7-69e00e97b5c1
spec:
  ingressClassName: nginx
  rules:

```

```

- host: www.wang.org
  http:
    paths:
      - backend:
          service:
            name: pod-test1
            port:
              number: 80
            path: /v1(/|$)(.*)
            pathType: Exact      #类型为Exact
      - backend:
          service:
            name: pod-test2
            port:
              number: 80
            path: /v2(/|$)(.*)
            pathType: Exact
status:
  loadBalancer:
    ingress:
      - ip: 10.0.0.99

[root@master1 ~]#kubectl get svc -n ingress-nginx
NAME                                TYPE                CLUSTER-IP      EXTERNAL-IP
PORT(S)                            AGE
ingress-nginx-controller           LoadBalancer       10.104.190.93   10.0.0.99
  80:30288/TCP,443:30485/TCP      2m55s
ingress-nginx-controller-admission ClusterIP            10.101.94.203   <none>
  443/TCP                        2m55s

#集群外测试访问
C:\Users\Wang>curl -H"host:www.wang.org" http://10.0.0.99/v1
kubernetes pod-test v0.1!! ClientIP: 10.244.2.26, ServerName: pod-test1-
5f8f7698d8-jrg5x, ServerIP: 10.244.1.64!
C:\Users\Wang>curl -H"host:www.wang.org" http://10.0.0.99/v2
kubernetes pod-test v0.2!! ClientIP: 10.244.2.26, ServerName: pod-test2-
f7d78c544-f415k, ServerIP: 10.244.1.70!

#如果externalTrafficPolicy为Cluster，可以访问任意集群节点的NodePort端口30288,如果
externalTrafficPolicy为Local，只能访问运行Controller的Pod所在节点的NodePort端口30288
C:\Users\Wang>curl -H"host:www.wang.org" http://10.0.0.201:30288/v1
kubernetes pod-test v0.2!! ClientIP: 10.244.2.31, ServerName: pod-test2-
f7d78c544-f415k, ServerIP: 10.244.1.70!
C:\Users\Wang>curl -H"host:www.wang.org" http://10.0.0.202:30288/v2
kubernetes pod-test v0.2!! ClientIP: 10.244.2.31, ServerName: pod-test2-
f7d78c544-f1892, ServerIP: 10.244.1.66!

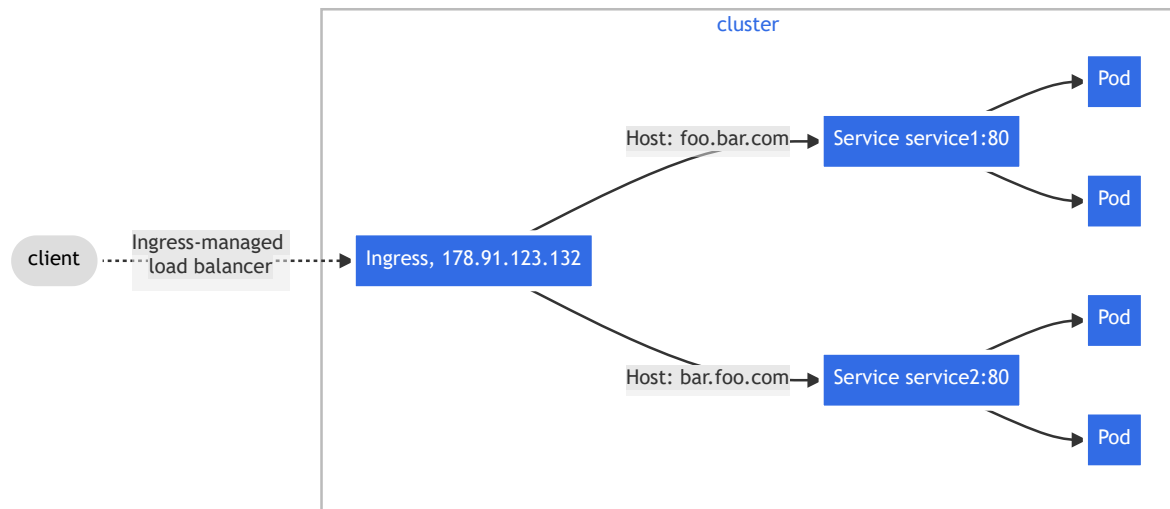
#支持子URL
C:\Users\Wang>curl -H"host:www.wang.org" http://10.0.0.99/v1/hostname
ServerName: pod-test1-5f8f7698d8-9vzgm
C:\Users\Wang>curl -H"host:www.wang.org" http://10.0.0.99/v2/hostname
ServerName: pod-test2-f7d78c544-f415k

#清理环境
[root@master1 ~]#kubectl delete ingress demo-ingress2

```



### 3.2.3 多域名



范例：命令式实现基于主机头的多虚拟主机

#环境准备：

#基于FQDN名称代理不同应用的请求时，需要事先准备好多个域名，并确保对这些域名的解析能够到达Ingress Controller

#对test1.wang.org的请求代理至service/pod-test1，对test2.wang.org请求代理至service/pod-test2

```
[root@master1 ~]#kubectl create ingress demo-ingress3 --
rule="test1.wang.org/*=pod-test1:80" --rule="test2.wang.org/*=pod-test2:80" --
class=nginx
```

```
[root@master1 ~]#kubectl get ingress
```

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
demo-ingress3	nginx	test1.wang.org,test2.wang.org		80	5s

```
[root@master1 ~]#kubectl get ingress demo-ingress3 -o yaml
```

```
apiVersion: networking.k8s.io/v1
```

```
kind: Ingress
```

```
metadata:
```

```
  creationTimestamp: "2021-04-02T10:41:03Z"
```

```
  generation: 1
```

```
  name: demo-ingress3
```

```
  namespace: default
```

```
  resourceVersion: "258984"
```

```
  uid: 8a5cbd1d-dcd2-4f1f-9dc9-9a8025e38b3a
```

```
spec:
```

```
  ingressClassName: nginx
```

```
  rules:
```

```
  - host: test1.wang.org
```

```
    http:
```

```
      paths:
```

```

- backend:
  service:
    name: pod-test1
    port:
      number: 80
  path: /
  pathType: Prefix #类型为Prefix, 原因为test1.wang.org/*有*号
- host: test2.wang.org
  http:
    paths:
      - backend:
        service:
          name: pod-test2
          port:
            number: 80
        path: /
        pathType: Prefix
status:
  loadBalancer: {}

#测试访问
C:\Users\Wang>curl -H"host:test1.wang.org" http://10.0.0.99
kubernetes pod-test v0.1!! ClientIP: 10.244.2.37, ServerName: pod-test1-5f8f7698d8-9vzgm, ServerIP: 10.244.1.60!
C:\Users\Wang>curl -H"host:test2.wang.org" http://10.0.0.99
kubernetes pod-test v0.2!! ClientIP: 10.244.2.37, ServerName: pod-test2-f7d78c544-f1892, ServerIP: 10.244.1.66!

#支持子URL
C:\Users\Wang>curl -H"host:test1.wang.org" http://10.0.0.99/hostname
ServerName: pod-test1-5f8f7698d8-kgv6n
C:\Users\Wang>curl -H"host:test2.wang.org" http://10.0.0.99/hostname
ServerName: pod-test2-f7d78c544-f415k

#清理环境
[root@master1 ~]#kubectl delete ingress demo-ingress3

```

## 3.2.4 HTTPS

### 3.2.4.1 命令式实现 HTTPS

范例: 命令式实现HTTPS

```

#基于TLS的Ingress要求事先准备好专用的“kubernetes.io/tls”类型的Secret资源对象
[root@master1 ~]#(umask 077; openssl genrsa -out www.wang.org.key 2048)

[root@master1 ~]#openssl req -new -x509 -key www.wang.org.key -out
www.wang.org.crt -subj /C=CN/ST=Beijing/L=Beijing/O=SRE/CN=www.wang.org -days
365

#创建Secret
[root@master1 ~]#kubectl create secret tls tls-wang --cert=./www.wang.org.crt --
key=./www.wang.org.key

[root@master1 ~]#kubectl get secrets

```

NAME	TYPE	DATA	AGE
tls-wang	kubernetes.io/tls	2	15m

```
[root@master1 ~]#kubectl describe secrets tls-wang
```

```
Name:          tls-wang
Namespace:     default
Labels:        <none>
Annotations:   <none>
```

```
Type: kubernetes.io/tls
```

```
Data
```

```
====
```

```
tls.crt: 1294 bytes
```

```
tls.key: 1700 bytes
```

#创建虚拟主机代理规则，同时将该主机定义为TLS类型，默认HTTP自动跳转至HTTPS

```
[root@master1 ~]#kubectl create ingress tls-demo-ingress --
rule='www.wang.org/*=pod-test1:80,tls=tls-wang' --class=nginx
```

#注意：启用tls后，该域名下的所有URI默认为强制将http请求利用308跳转至https，若不希望使用该跳转功能，可以使用如下注解选项

#方法1

--annotation nginx.ingress.kubernetes.io/ssl-redirect=false, 即如下形式

```
[root@master1 ~]#kubectl create ingress tls-demo-ingress --
rule='www.wang.org/*=pod-test1:80,tls=tls-wang' --class=nginx --annotation
nginx.ingress.kubernetes.io/ssl-redirect=false
```

#方法2

```
[root@master1 ~]#kubectl edit ingress tls-demo-ingress
```

```
metadata:
```

```
  annotations: #添加两行
```

```
    nginx.ingress.kubernetes.io/ssl-redirect: "false"
```

#查看

```
[root@master1 ~]#kubectl get ingress tls-demo-ingress -o yaml
```

```
apiVersion: networking.k8s.io/v1
```

```
kind: Ingress
```

```
metadata:
```

```
  creationTimestamp: "2021-04-02T11:15:46Z"
```

```
  generation: 1
```

```
  name: tls-demo-ingress
```

```
  namespace: default
```

```
  resourceVersion: "262472"
```

```
  uid: ecff3bd3-da04-4df0-8b79-7a8e23962ca2
```

```
spec:
```

```
  ingressClassName: nginx
```

```
  rules:
```

```
  - host: www.wang.org
```

```
    http:
```

```
      paths:
```

```
      - backend:
```

```
          service:
```

```
            name: pod-test1
```

```
            port:
```

```
              number: 80
```

```
          path: /
```

```
          pathType: Prefix
```

```
  tls:
```

```
  - hosts:
```

```
- www.wang.org
secretName: tls-wang
status:
  loadBalancer:
    ingress:
      - ip: 10.0.0.99
```

#集群外客户端测试访问

```
C:\Users\wang>curl -H"host: www.wang.org" http://10.0.0.99/
<html>
<head><title>308 Permanent Redirect</title></head>
<body>
<center><h1>308 Permanent Redirect</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

```
C:\Users\wang>curl -IH"host: www.wang.org" http://10.0.0.99/
HTTP/1.1 308 Permanent Redirect
Date: Thu, 15 Jun 2022 03:11:04 GMT
Content-Type: text/html
Content-Length: 164
Connection: keep-alive
Location: https://www.wang.org
```

#注意:修改为https方式访问

```
C:\Users\wang>curl -k -H"host: www.wang.org" https://10.0.0.99/
kubernetes pod-test v0.1!! ClientIP: 10.244.2.37, ServerName: pod-test1-
5f8f7698d8-jrg5x, ServerIP: 10.244.1.64!
```

#查看证书有效期

#注意: openssl命令的版本过低将无法看到过期时间, 比如:Centos8

```
[root@ubuntu2204 ~]#openssl s_client www.wang.org:443
CONNECTED(00000003)
depth=0 C = CN, ST = Beijing, L = Beijing, O = SRE, CN = www.wang.org
verify error:num=18:self-signed certificate
verify return:1
depth=0 C = CN, ST = Beijing, L = Beijing, O = SRE, CN = www.wang.org
verify return:1
---
Certificate chain
 0 s:C = CN, ST = Beijing, L = Beijing, O = SRE, CN = www.wang.org
  i:C = CN, ST = Beijing, L = Beijing, O = SRE, CN = www.wang.org
  a:PKKEY: rsaEncryption, 2048 (bit); sigalg: RSA-SHA256
  v:NotBefore: Jan  9 04:01:09 2025 GMT; NotAfter: Jan  7 04:01:09 2035 GMT
```

#清理环境

```
[root@master1 ~]#kubectl delete ingress tls-demo-ingress
```

#修改为不自动跳转https

```
[root@master1 ~]#kubectl create ingress tls-demo-ingress --
rule='www.wang.org/*=pod-test1:80,tls=tls-wang' --class=nginx --annotation
nginx.ingress.kubernetes.io/ssl-redirect=false
```

```
[root@master1 ~]#kubectl get ingress tls-demo-ingress
```

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
tls-demo-ingress	nginx	www.wang.org	10.0.0.99	80, 443	53s

```
[root@master1 ~]#kubectl get ingress tls-demo-ingress -o yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    nginx.ingress.kubernetes.io/ssl-redirect: "false"    #不自动跳转HTTPS
  creationTimestamp: "2021-04-20T10:13:26Z"
  generation: 1
  name: tls-demo-ingress
  namespace: default
  resourceVersion: "677174"
  uid: a2e8e19e-fc10-4e80-ad68-ac7c2ab038c6
spec:
  ingressClassName: nginx
  rules:
  - host: www.wang.org
    http:
      paths:
      - backend:
          service:
            name: pod-test1
            port:
              number: 80
          path: /
          pathType: Prefix
    tls:
    - hosts:
      - www.wang.org
      secretName: tls-wang
status:
  loadBalancer:
    ingress:
    - ip: 10.0.0.99

#测试从集群外访问,发现不再自动跳转HTTPS
C:\Users\Wang>curl -H"host: www.wang.org" http://10.0.0.99/
kubernetes pod-test v0.1!! ClientIP: 10.244.2.31, ServerName: pod-test1-
5f8f7698d8-xzvdF, ServerIP: 10.244.2.32!

C:\Users\Wang>curl -kH"host: www.wang.org" https://10.0.0.99/
kubernetes pod-test v0.1!! ClientIP: 10.244.2.31, ServerName: pod-test1-
5f8f7698d8-44vvk, ServerIP: 10.244.3.14!
```

### 3.2.4.2 证书更新

HTTPS 的证书的有效期一般为1年,到期前需要提前更新证书

范例: 证书更新

```
#重新颁发证书
[root@master1 ~]#(umask 077; openssl genrsa -out wang.key 2048)
[root@master1 ~]#openssl req -new -x509 -key wang.key -out wang.crt -subj
/C=CN/ST=Beijing/L=Beijing/O=SRE/CN=www.wang.org -days 3650

#方法1:
#在线修改证书配置,需要提前先将新证书文件用base64编码并删除换行符
```

```

[root@master1 ~]#cat wang.crt |base64 | tr -d '\n'
[root@master1 ~]#cat wang.key |base64 | tr -d '\n'
#上面生成的内容替换下面命令的内容,立即生效
[root@master1 ~]#kubectl edit secrets tls-wang

#方法2
#删除旧证书配置
[root@master1 ~]#kubectl delete secrets tls-wang
#创建新证书配置
[root@master1 ~]#kubectl create secret tls tls-wang --cert=./wang.crt --
key=./wang.key

#集群外验证证书是否生效
[root@ubuntu2204 ~]#echo 10.0.0.99 www.wang.org >> /etc/hosts

#注意: openssl命令的版本过低将无法看到过期时间,比如:Centos8
[root@ubuntu2204 ~]#openssl s_client www.wang.org:443
CONNECTED(00000003)
depth=0 C = CN, ST = Beijing, L = Beijing, O = SRE, CN = www.wang.org
verify error:num=18:self-signed certificate
verify return:1
depth=0 C = CN, ST = Beijing, L = Beijing, O = SRE, CN = www.wang.org
verify return:1
---
Certificate chain
 0 s:C = CN, ST = Beijing, L = Beijing, O = SRE, CN = www.wang.org
  i:C = CN, ST = Beijing, L = Beijing, O = SRE, CN = www.wang.org
  a:PKKEY: rsaEncryption, 2048 (bit); sigalg: RSA-SHA256
  v:NotBefore: Jun 15 03:18:44 2021 GMT; NotAfter: Jun 12 03:18:44 2031 GMT
.....

#清理环境
[root@master1 ~]#kubectl delete ingress tls-demo-ingress
[root@master1 ~]#kubectl delete secrets tls-wang

```

## 4 Ingress 声明式实现

### 4.1 声明式实现说明

基于命令方式格式功能有限,且不利于后续的重复使用,工作中更多的使用声明式实现Ingress

在实际的工作中,可能会基于域名访问,也可能会基于不同的功能服务以子路径的方式来进行访问,以及与https相关的访问。

注意: k8s-v1.22之前后版本的声明式的yaml格式有所不同

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

官方配置参考

<https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/annotations/>

配置文件解析

apiVersion: networking.k8s.io/v1	# 资源所属的API群组和版本
kind: Ingress	# 资源类型标识符
metadata:	# 元数据
name <string>	# 资源名称
annotations:	# 资源注解，v1beta1使用下面的注解来指定要
解析该资源的控制器类型	
kubernetes.io/ingress.class: <string>	# 适配的Ingress控制器类别,便于多ingress组
件场景下,挑选针对的类型	
namespace <string>	# 名称空间
spec:	
rules <[]Object>	# Ingress规则列表,也就是http转发时候用到
的 url关键字	
- host <string>	# 虚拟主机的FQDN,支持“*”前缀通配,不支持
IP, 不支持指定端口	
http <Object>	
paths <[]Object>	# 虚拟主机PATH定义的列表,由path和backend
组成	
- path <string>	# 流量匹配的HTTP PATH,必须以/开头
pathType <string>	# 支持Exact、Prefix和
ImplementationSpecific, 必选	
backend <Object>	# 匹配到的流量转发到的目标后端
resource <Object>	# 引用的同一名称空间下的资源,与下面两个字段
互斥	
service <object>	# 关联的后端Service对象
name <string>	# 后端Service的名称
port <object>	# 后端Service上的端口对象
name <string>	# 端口名称
number <integer>	# 端口号
tls <[]Object>	# TLS配置,用于指定上rules中定义的哪些host需要工作
HTTPS模式	
- hosts <[]string>	# 使用同一组证书的主机名称列表
secretName <string>	# 保存于数字证书和私钥信息的secret资源名称,用于主机
认证	
backend <Object>	# 默认backend的定义,可嵌套字段及使用格式跟rules字
段中的相同	
ingressClassName <string>	# ingress类名称,用于指定适配的控制器,类似于注解的
功能,未来代替annotations	



Annotations - Ingress-nginx Cor
+
https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/a...
Annotations
Search
kubernetes/ingress-nginx
15.1k Stars · 7.9k Forks

**User guide**
**NGINX Configuration**
Introduction
Basic usage
Annotations
ConfigMap
Custom NGINX template
Log format
Command line arguments
Custom errors
Default backend
Exposing TCP and UDP services
Exposing FCGI services
Regular expressions in paths
External Articles
Miscellaneous
Prometheus and Grafana installation
Multiple Ingress controllers
TLS/HTTPS
**Third party addons**
ModSecurity Web Application Firewall
OpenTracing
OpenTelemetry

Name	type
<a href="#">nginx.ingress.kubernetes.io/app-root</a>	string
<a href="#">nginx.ingress.kubernetes.io/affinity</a>	cookie
<a href="#">nginx.ingress.kubernetes.io/affinity-mode</a>	"balanced" or "persistent"
<a href="#">nginx.ingress.kubernetes.io/affinity-canary-behavior</a>	"sticky" or "legacy"
<a href="#">nginx.ingress.kubernetes.io/auth-realm</a>	string
<a href="#">nginx.ingress.kubernetes.io/auth-secret</a>	string
<a href="#">nginx.ingress.kubernetes.io/auth-secret-type</a>	string
<a href="#">nginx.ingress.kubernetes.io/auth-type</a>	"basic" or "digest"
<a href="#">nginx.ingress.kubernetes.io/auth-tls-secret</a>	string
<a href="#">nginx.ingress.kubernetes.io/auth-tls-verify-depth</a>	number
<a href="#">nginx.ingress.kubernetes.io/auth-tls-verify-client</a>	string
<a href="#">nginx.ingress.kubernetes.io/auth-tls-error-page</a>	string
<a href="#">nginx.ingress.kubernetes.io/auth-tls-pass-certificate-to-upstream</a>	"true" or "false"
<a href="#">nginx.ingress.kubernetes.io/auth-tls-match-cn</a>	string
<a href="#">nginx.ingress.kubernetes.io/auth-url</a>	string
<a href="#">nginx.ingress.kubernetes.io/auth-cache-key</a>	string

**Table of contents**
Canary
Rewrite
Session Affinity
Cookie affinity
Authentication
Custom NGINX upstream hashing
Custom NGINX load balancing
Custom NGINX upstream vhost
Client Certificate Authentication
Backend Certificate Authentication
Configuration snippet
Custom HTTP Errors
Default Backend
Enable CORS
HTTP2 Push Preload.
Server Alias
Server snippet
Client Body Buffer Size
External Authentication
Global External Authentication
Rate Limiting
Global Rate Limiting
Permanent Redirect
Permanent Redirect Code
Temporal Redirect

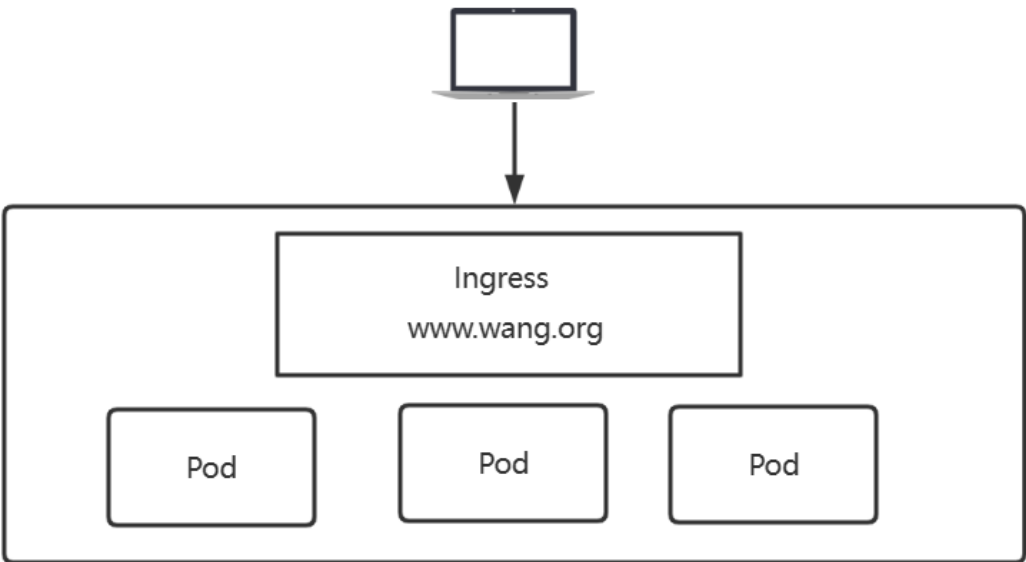
https://kubernetes.io/zh-cn/docs/concepts/services-networking/ingress/#%E7%A4%BA%E4%BE%8B

示例

类型	路径	请求路径	匹配与否?
Prefix	/	(所有路径)	是
Exact	/foo	/foo	是
Exact	/foo	/bar	否
Exact	/foo	/foo/	否
Exact	/foo/	/foo	否
Prefix	/foo	/foo, /foo/	是
Prefix	/foo/	/foo, /foo/	是
Prefix	/aaa/bb	/aaa/bbb	否
Prefix	/aaa/bbb	/aaa/bbb	是
Prefix	/aaa/bbb/	/aaa/bbb	是, 忽略尾部斜线
Prefix	/aaa/bbb	/aaa/bbb/	是, 匹配尾部斜线
Prefix	/aaa/bbb	/aaa/bbb/ccc	是, 匹配子路径
Prefix	/aaa/bbb	/aaa/bbbxyz	否, 字符串前缀不匹配
Prefix	/, /aaa	/aaa/ccc	是, 匹配 /aaa 前缀
Prefix	/, /aaa, /aaa/bbb	/aaa/bbb	是, 匹配 /aaa/bbb 前缀
Prefix	/, /aaa, /aaa/bbb	/ccc	是, 匹配 / 前缀
Prefix	/aaa	/ccc	否, 使用默认后端
混合	/foo (Prefix), /foo (Exact)	/foo	是, 优选 Exact 类型

## 4.2 声明式实现案例

### 4.2.1 单域名案例



nginx controller 可以实现的基于域名的访问

范例：单域名支持子URL

#准备后端服务所需的资源

```
[root@master1 ~]#cat ingress-deployment-svc.yaml
```

```
apiVersion: apps/v1
```

```
kind: Deployment
```

```
metadata:
```

```
  name: deployment-test
```

```
spec:
```

```
  replicas: 3
```

```
  selector:
```

```
    matchLabels:
```

```
      app: pod-test
```

```
  template:
```

```
    metadata:
```

```
      labels:
```

```
        app: pod-test
```

```
    spec:
```

```
      containers:
```

```
        - name: pod-test
```

```
          image: registry.cn-beijing.aliyuncs.com/wangxiaochun/pod-test:v0.1
```

```
          imagePullPolicy: IfNotPresent
```

```
          ports:
```

```
            - containerPort: 80
```

```
              name: http
```

```
---
```

```
apiVersion: v1
```

```
kind: Service
```

```
metadata:
```

```
  name: deployment-service
```

```
spec:
```

```
  selector:
```

```
    app: pod-test
```

```
  ports:
```

```
    - name: http
```

```
      port: 80
```

```
      targetPort: 80
```

#应用资源文件

```
[root@master1 ~]#kubectl apply -f ingress-deployment-svc.yaml
```

#查看效果

```
[root@master1 ~]#kubectl get deployments,svc,po
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/deployment-test	3/3	3	3	105s

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
service/deployment-service	ClusterIP	10.99.228.134	<none>	80/TCP
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP

NAME	READY	STATUS	RESTARTS	AGE
pod/deployment-test-555f594785-9mmbq	1/1	Running	0	105s

pod/deployment-test-555f594785-16v98	1/1	Running	0	105s
pod/deployment-test-555f594785-rfhcz	1/1	Running	0	105s

```
[root@master1 ~]#kubectl get po -o wide
```

NAME	READY	STATUS	RESTARTS	AGE	IP
deployment-test-555f594785-9mmbq	1/1	Running	0	6m32s	
10.244.5.38 node3.wang.org	<none>	<none>			
deployment-test-555f594785-16v98	1/1	Running	0	6m32s	
10.244.3.66 node1.wang.org	<none>	<none>			
deployment-test-555f594785-rfhcz	1/1	Running	0	6m32s	
10.244.3.67 node1.wang.org	<none>	<none>			

#定义创建ingress资源文件，和Controller无关，使用统一格式

```
[root@master1 ~]#cat ingress-http-test.yaml
```

```
apiVersion: networking.k8s.io/v1 #v1.19版之前使用networking.k8s.io/v1beta1,
且下面内容格式也所有不同
```

```
kind: Ingress
```

```
metadata:
```

```
  name: ingress-test
```

```
  annotations:
```

```
    # kubernetes.io/ingress.class: "nginx" #旧版指定nginx做为ingress controller, 支持其它controller,如: traefik, k8s-v1.32.0不再支持
```

```
spec:
```

```
  ingressClassName: nginx #新版建议使用此项指定controller类型
```

```
  rules:
```

```
    - host: www.wang.org #域名配置,如果不写相当于匹配 *,支持格式:*.wang.org
```

```
      http:
```

```
        paths: #相当于nginx的location配置块,同一个host可以配置多个path
```

```
          - path: / #将www.wang.org的请求代理至后端的/
```

```
            pathType: Prefix #表示以 / 开始即可
```

```
            backend:
```

```
              service:
```

```
                name: deployment-service #service的名称
```

```
                port:
```

```
                  number: 80 #service的Port
```

#属性解析：注释信息，对于不同的ingress的内容是不一样的,对于ingress来说，最重要的就是rules了，这里的hosts是一个自定义的域名

#应用资源文件

```
[root@master1 ~]#kubectl apply -f ingress-http-test.yaml
```

#查看效果

```
[root@master1 ~]#kubectl get ingress
```

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
ingress-test	<none>	www.wang.org	10.0.0.104	80	32s

```
[root@master1 ~]#kubectl describe ingress ingress-test
```

```
Name: ingress-test
```

```
Namespace: default
```

```
Address: 10.0.0.104
```

```
Default backend: default-http-backend:80 (<error: endpoints "default-http-backend" not found>)
```

```
Rules:
```

```
  Host      Path      Backends
```

```
  ----      -
```

```
  www.wang.org
```

```
        / deployment-service:80
(10.244.3.66:80,10.244.3.67:80,10.244.5.38:80)
Annotations:      kubernetes.io/ingress.class: nginx
Events:
```

Type	Reason	Age	From	Message
Normal	Sync	45s (x2 over 55s)	nginx-ingress-controller	Scheduled for sync

#结果显示: ingress 将后端的deployment-service:80 的样式自动与 ingress 的 url(/) 关联在一起

#为externalIPs的ip地址做一个域名绑定

```
C:\Users\Wang>echo '10.0.0.99 www.wang.org' >> /etc/hosts
```

#访问效果,注意:需要提前做名称解析

```
C:\Users\Wang>curl www.wang.org
kubernetes pod-test v0.1!! ClientIP: 10.244.3.65, ServerName: deployment-test-555f594785-l6v98, ServerIP: 10.244.3.66!
C:\Users\Wang>curl www.wang.org
kubernetes pod-test v0.1!! ClientIP: 10.244.3.65, ServerName: deployment-test-555f594785-rfhcz, ServerIP: 10.244.3.67!
C:\Users\Wang>curl www.wang.org
kubernetes pod-test v0.1!! ClientIP: 10.244.3.65, ServerName: deployment-test-555f594785-9mbq, ServerIP: 10.244.5.38!
```

#非域名方式进行访问

```
C:\Users\Wang>curl 10.0.0.99
<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

#结果显示: 通过域名可以正常的访问后端的服务,而非域名访问方式受到了限制

```
C:\Users\Wang>curl -H "Host:www.wang.org" http://10.0.0.99
kubernetes pod-test v0.1!! ClientIP: 10.244.2.5, ServerName: deployment-test-95c58b447-jg6gz, ServerIP: 10.244.2.6!
```

#如果type的值为LoadBalancer或者NodePort,也可直接访问运行Controller Pod的节点的IP:32453也可以访问

```
C:\Users\Wang>curl -H "Host:www.wang.org" http://10.0.0.106:32453
kubernetes pod-test v0.1!! ClientIP: 10.244.3.65, ServerName: deployment-test-555f594785-l6v98, ServerIP: 10.244.3.66!
```

#结果显示: 通过自定义携带域名的方式可以正常访问。

#进入到ingress的控制器端,查看配置效果

```
[root@master1 ~]#kubectl -n ingress-nginx exec -it ingress-nginx-controller-548b8cb9b8-tsxpl -- /bin/bash -c 'grep example nginx.conf'
## start server www.wang.org
server_name www.wang.org ;
## end server www.wang.org
```

```
[root@master1 ~]#kubectl -n ingress-nginx exec -it ingress-nginx-controller-548b8cb9b8-tsxpl -- cat nginx.conf
```

#结果显示：这里直接将我们提前定义的域名信息作为信息的入口了

#可以通过haproxy配置实现访问,先查看ingress-nginx-controller对应service的NodePort节点对应的Port

```
[root@master1 ~]#kubectl get svc -n ingress-nginx
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP
ingress-nginx-controller	LoadBalancer	10.98.85.194	10.0.0.99
ingress-nginx-controller-admission	ClusterIP	10.106.123.179	<none>

#基于集群外的Haproxy和NodePort的SVC实现

```
[root@ha1 ~]#vim /etc/haproxy/haproxy.cfg
```

#添加下面行

```
listen www.wang.org-80
  bind 10.0.0.100:80
  mode http
  server node1 10.0.0.103:32453 check inter 3s fall 3 rise 3
  server node2 10.0.0.104:32453 check inter 3s fall 3 rise 3
  server node3 10.0.0.105:32453 check inter 3s fall 3 rise 3
```

```
[root@ha1 ~]#systemctl restart haproxy
```

#修改hosts

```
[root@master1 ~]#vim /etc/hosts
```

```
10.0.0.100 k8s k8s.wang.org www.wang.org
```

#再次访问

```
[root@master1 ~]#curl www.wang.org
```

```
kubernetes pod-test v0.1!! ClientIP: 10.244.3.65, ServerName: deployment-test-555f594785-9mmbq, ServerIP: 10.244.5.38!
```

```
[root@master1 ~]#curl www.wang.org
```

```
kubernetes pod-test v0.1!! ClientIP: 10.244.3.65, ServerName: deployment-test-555f594785-l6v98, ServerIP: 10.244.3.66!
```

```
[root@master1 ~]#curl www.wang.org
```

```
kubernetes pod-test v0.1!! ClientIP: 10.244.3.65, ServerName: deployment-test-555f594785-rfhcz, ServerIP: 10.244.3.67!
```

#清理环境

```
[root@master1 ~]#kubectl delete -f yaml/ingress-http-test.yaml
```

## 4.2.2 获取真实客户端IP

范例：获取真实客户端IP，需要通过loadBalancer的SVC实现才支持，ExternalIP不支持

#准备环境

```
[root@master1 ~]#kubectl create deployment myapp --image registry.cn-beijing.aliyuncs.com/wangxiaochun/nginx:1.20.0
```

```
[root@master1 ~]#kubectl create svc clusterip myapp --tcp 80
```

#Ingress配置

```
[root@master1 ~]#cat ingress-http-real-ip.yaml
```

```
apiVersion: networking.k8s.io/v1
```

```
kind: Ingress
```

```
metadata:
```

```
  generation: 1
```

```
  name: ingress-myapp
```

```

namespace: default
annotations:
  nginx.ingress.kubernetes.io/enable-real-ip: "true" #允许IP透传,此为默认值,可以不
添加此行
spec:
  ingressClassName: nginx
  rules:
  - host: www.wang.org
    http:
      paths:
      - backend:
          service:
            name: myapp
            port:
              number: 80
        path: /
        pathType: Prefix

```

#创建Ingress资源

```
[root@master1 ~]#kubectl apply -f ingress-http-real-ip.yaml
```

#查看Ingress资源

```
[root@master1 ~]#kubectl describe ingress ingress-myapp
```

```

Name:          ingress-myapp
Labels:         <none>
Namespace:     default
Address:        10.0.0.99
Ingress Class:  nginx
Default backend: <default>
Rules:
  Host          Path    Backends
  ----          -
  www.wang.org  /      myapp:80 (10.244.2.194:80)
Annotations:    nginx.ingress.kubernetes.io/enable-real-ip: true
Events:
  Type    Reason    Age              From              Message
  ----    -
  Normal  Sync      3m26s (x2 over 3m35s)  nginx-ingress-controller  Scheduled for sync

```

#查看Ingress资源

```
[root@master1 ~]#kubectl get ingress ingress-myapp -o yaml
```

```

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"networking.k8s.io/v1","kind":"Ingress","metadata":
{"annotations":{"nginx.ingress.kubernetes.io/enable-real-
ip":"true"},"generation":1,"name":"ingress-myapp","namespace":"default"},"spec":
{"ingressClassName":"nginx","rules":[{"host":"www.wang.org","http":{"paths":
[{"backend":{"service":{"name":"myapp","port":
{"number":80}}},"path":"/","pathType":"Prefix"}]}]}
    nginx.ingress.kubernetes.io/enable-real-ip: "true"
  creationTimestamp: "2022-05-28T02:15:27Z"
  generation: 1
  name: ingress-myapp

```

```

namespace: default
resourceVersion: "1511049"
uid: ad8c3a43-e400-47dd-87da-2241ac5237a7
spec:
  ingressClassName: nginx
  rules:
  - host: www.wang.org
    http:
      paths:
      - backend:
          service:
            name: myapp
            port:
              number: 80
        path: /
        pathType: Prefix
status:
  loadBalancer:
    ingress:
    - ip: 10.0.0.99

```

#查看ingress nginx 配置

```
[root@master1 ~]#kubectl get pod -n ingress-nginx
```

NAME	READY	STATUS	RESTARTS
AGE			
ingress-nginx-admission-patch-7k94z	0/1	Completed	0
16h			
ingress-nginx-controller-6fc8fc9c5b-7tkzz	1/1	Running	1 (3h29m ago)
16h			

```
[root@master1 ~]#kubectl exec -n ingress-nginx ingress-nginx-controller-6fc8fc9c5b-7tkzz -- nginx -T |grep 'proxy_set_header X-Forwarded-For'
proxy_set_header X-Forwarded-For $remote_addr;
```

```
[root@master1 ~]#kubectl exec -n ingress-nginx ingress-nginx-controller-6fc8fc9c5b-7tkzz -- nginx -T |grep 'proxy_set_header X-Real-IP'
proxy_set_header X-Real-IP $remote_addr;
```

#从集群内访问

```
[root@master1 ~]#curl -H"host: www.wang.org" 10.0.0.99
welcome to Nginx web Site ! Nginx Version: 1.20.0
```

#集群外访问

```
C:\Users\Wang>curl -H"host: www.wang.org" 10.0.0.99
welcome to Nginx web Site ! Nginx Version: 1.20.0
```

```
[root@rocky8 ~]#curl -H"host: www.wang.org" 10.0.0.99
welcome to Nginx web Site ! Nginx Version: 1.20.0
```

#查看Pod的名称

```
[root@master1 ~]#kubectl get pod
```

NAME	READY	STATUS	RESTARTS	AGE
myapp-6c88c9948-v9htm	1/1	Running	0	20m

#查看Pod的访问日志格式

```
[root@master1 ~]#kubectl exec -it myapp-6c88c9948-v9htm -- nginx -T |grep -A3 log_format
log_format main '$remote_addr - $remote_user [$time_local] "$request" '
```



```
'$status $body_bytes_sent "$http_referer" '
'"$http_user_agent" "$http_x_forwarded_for";
```

#查看ingress-controller的地址

```
[root@master1 ~]#kubectl get pod -n ingress-nginx ingress-nginx-controller-6fc8fc9c5b-7tkzz -o wide
```

NAME		READY	STATUS	RESTARTS	AGE		
IP	NODE	NOMINATED	NODE	READINESS	GATES		
ingress-nginx-controller-6fc8fc9c5b-7tkzz	10.244.1.182	node1.wang.org	<none>	1/1	Running	1 (89m ago)	14h

#查看访问日志可看到\$http\_x\_forwarded\_for字段记录集群外客户端的真实地址，\$remote\_addr字段记录的是ingress-controller的地址

```
[root@master1 ~]#kubectl logs -f myapp-6c88c9948-v9htm
```

```
10.244.1.182 - - [28/May/2022:02:17:52 +0000] "GET / HTTP/1.1" 200 50 "-"
"curl/7.81.0" "10.244.0.0"
```

```
10.244.1.182 - - [28/May/2022:02:17:53 +0000] "GET / HTTP/1.1" 200 50 "-"
"curl/8.0.1" "10.0.0.1"
```

```
10.244.1.182 - - [28/May/2022:02:23:10 +0000] "GET / HTTP/1.1" 200 50 "-"
"curl/7.61.1" "10.0.0.8"
```

## 4.2.3 单域名多URL案例

```
https://kubernetes.io/docs/concepts/services-networking/ingress/
```

Ingress | Kubernetes

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

kubernetes

Documentation
Kubernetes Blog
Training
Partners
Community
Case Studies
Versions
English

Q Search

Home
Getting started
Concepts

Overview

Cluster Architecture

Containers

Workloads

Services, Load Balancing, and Networking

Service
Ingress
Ingress Controllers
EndpointSlices
Network Policies
DNS for Services and Pods
IPv4/IPv6 dual-stack
Topology Aware Routing
Networking on Windows
Service ClusterIP allocation
Service Internal Traffic Policy
Topology-aware traffic routing with topology keys

Storage

Configuration

Security

Policies

Scheduling, Preemption and Eviction

Cluster Administration

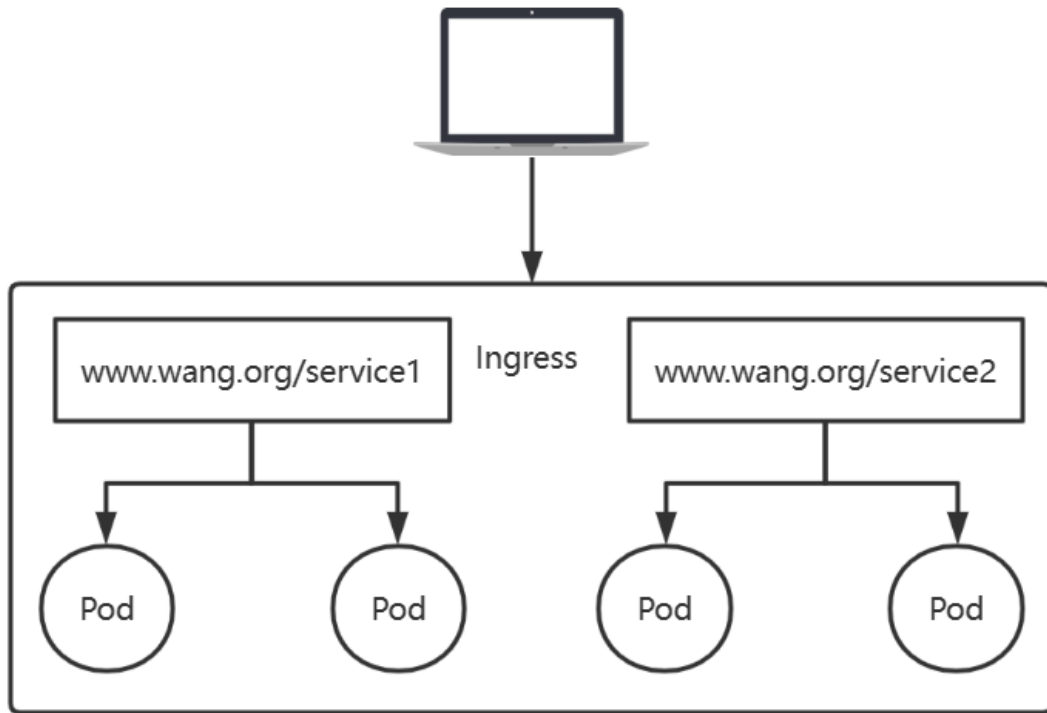
Windows in

Examples

Kind	Path(s)	Request path(s)	Matches?
Prefix	/	(all paths)	Yes
Exact	/foo	/foo	Yes
Exact	/foo	/bar	No
Exact	/foo	/foo/	No
Exact	/foo/	/foo	No
Prefix	/foo	/foo , /foo/	Yes
Prefix	/foo/	/foo , /foo/	Yes
Prefix	/aaa/bb	/aaa/bbb	No
Prefix	/aaa/bbb	/aaa/bbb	Yes
Prefix	/aaa/bbb/	/aaa/bbb	Yes, ignores trailing slash
Prefix	/aaa/bbb	/aaa/bbb/	Yes, matches trailing slash
Prefix	/aaa/bbb	/aaa/bbb/ccc	Yes, matches subpath
Prefix	/aaa/bbb	/aaa/bbbxyz	No, does not match string prefix
Prefix	/ , /aaa	/aaa/ccc	Yes, matches /aaa prefix
Prefix	/ , /aaa , /aaa/bbb	/aaa/bbb	Yes, matches /aaa/bbb prefix
Prefix	/ , /aaa , /aaa/bbb	/ccc	Yes, matches / prefix
Prefix	/aaa	/ccc	No, uses default backend
Mixed	/foo (Prefix), /foo (Exact)	/foo	Yes, prefers Exact

Multiple matches

In some cases, multiple paths within an Ingress will match a request. In those cases precedence will be given first to the longest matching path. If two paths are still equally matched, precedence will be given to paths with an exact path type over prefix path type.



Simple fanout: 在同一个FQDN下通过不同的URI完成不同应用间的流量分发

- 基于单个虚拟主机接收多个应用的流量
- 常用于将流量分发至同一个应用下的多个不同子应用，同一个应用内的流量由调度算法分发至该应用的各后端端点
- 不需要为每个应用配置专用的域名

范例：环境准备两个HTTP应用

#如果前面的资源已删除，重新应用上面小节的资源文件生成deployment和对应的SVC  
#访问 [www.wang.org/flask](http://www.wang.org/flask)的时候，返回flask的结果  
#访问 [www.wang.org/nginx](http://www.wang.org/nginx)的时候，返回nginx的结果

```
[root@master1 ~]#cat ingress-deployment-svc.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: deployment-test
spec:
  replicas: 3
  selector:
    matchLabels:
      app: pod-test
  template:
    metadata:
      labels:
        app: pod-test
    spec:
      containers:
        - name: pod-test
          image: registry.cn-beijing.aliyuncs.com/wangxiaochun/pod-test:v0.1
          imagePullPolicy: IfNotPresent
          ports:
            - containerPort: 80
```

```

        name: http
---
apiVersion: v1
kind: Service
metadata:
  name: deployment-service
spec:
  selector:
    app: pod-test
  ports:
    - name: http
      port: 80
      targetPort: 80

```

```
[root@master1 ~]#kubectl apply -f yaml/ingress-deployment-svc.yaml
```

#再添加一个nginx的服务，定义资源文件

```
[root@master1 ~]#cat ingress-deployment-nginx.yaml
```

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: deployment-nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx-test
  template:
    metadata:
      labels:
        app: nginx-test
    spec:
      containers:
        - name: nginx-test
          image: registry.cn-beijing.aliyuncs.com/wangxiaochun/nginx:1.20.0
          imagePullPolicy: IfNotPresent
          ports:
            - containerPort: 80
              name: nginx
---
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx-test
  ports:
    - name: nginx
      port: 80
      targetPort: 80

```

---

```

apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx-test
  ports:
    - name: nginx
      port: 80
      targetPort: 80

```

#配置解析:为了避免多个service使用同一个后端端口，这里一定要为service的端口命名

#应用资源文件

```
[root@master1 ~]#kubectl apply -f ingress-deployment-nginx.yaml
```

#查看效果

```
[root@master1 ~]#kubectl get deployment,svc
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/deployment-nginx	3/3	3	3	88s
deployment.apps/deployment-test	3/3	3	3	73m

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
AGE				
service/deployment-service	ClusterIP	10.99.228.134	<none>	80/TCP
73m				
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP
5d21h				
service/nginx-service	ClusterIP	10.99.61.212	<none>	80/TCP
88s				

### 4.2.3.1 单域名多URL不支持子URL

范例: 单域名多URL, 不支持子URL

```
#清单文件
[root@master1 ~]#cat ingress-http-mul-url.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-mul-url
  annotations:
    #kubernetes.io/ingress.class: "nginx"
    nginx.ingress.kubernetes.io/rewrite-target: / #默认会转发给后端时会带有URL, 添加此行, 表示转发时删除后面的URL
spec:
  ingressClassName: nginx #新版建议使用此项指定controller类型
  rules:
  - host: www.wang.org
    http:
      paths:
      - path: /flask #指定访问的URL
        pathType: Prefix #表示以/flask为开始即可
        backend:
          service:
            name: deployment-service #指定对应service的名称
            port:
              name: http #匹配对应service中的端口name: http
              #number: 80
      - path: /nginx #指定访问的URL
        pathType: Prefix #表示以/nginx为开始即可
        backend:
          service:
            name: nginx-service #指定对应service的名称
            port:
              name: nginx #匹配对应service中的端口name: nginx
              #number: 80

#注意事项:
#默认转发给后端服务时会url也同时转发, 而后端服务有可能不存在此URL, 所以需要在后端url转发的时候, 取消转发关键字。
#方法就是, 在annotation中添加一个重写的规则nginx.ingress.kubernetes.io/rewrite-target: / 即所有的请求把ingress匹配到的url关键字清除掉

#应用资源文件
```

```
[root@master1 ~]#kubectl apply -f ingress-http-mul-url.yaml

#查看效果
[root@master1 ~]#kubectl get ingress
NAME                CLASS      HOSTS          ADDRESS        PORTS    AGE
ingress-mul-url     <none>    www.wang.org   10.0.0.104     80       68s

[root@master1 ~]#kubectl describe ingress
Name:                ingress-mul-url
Namespace:           default
Address:             10.0.0.104
Default backend:     default-http-backend:80 (<error: endpoints "default-http-backend" not found>)
Rules:
  Host              Path  Backends
  ----              -
  www.wang.org
                    /flask  deployment-service:http
(10.244.3.66:80,10.244.3.67:80,10.244.5.38:80)
                    /nginx  nginx-service:nginx
(10.244.3.68:80,10.244.4.51:80,10.244.5.39:80)
Annotations:         kubernetes.io/ingress.class: nginx
                    nginx.ingress.kubernetes.io/rewrite-target: /
Events:
  Type    Reason    Age           From                      Message
  ----    -
  Normal  Sync      37s (x2 over 73s)  nginx-ingress-controller  Scheduled for sync

#修改hosts文件(配合前面的hproxy实现)
[root@master1 ~]#vim /etc/hosts
10.0.0.99 k8s k8s.wang.org www.wang.org

#浏览器访问效果
[root@master1 ~]#curl www.wang.org/flask
kubernetes pod-test v0.1!! ClientIP: 10.244.3.65, ServerName: deployment-test-555f594785-l6v98, ServerIP: 10.244.3.66!
[root@master1 ~]#curl www.wang.org/nginx
welcome to Nginx Web Site ! Nginx Version: 1.20.0

#清理环境
[root@master1 ~]#kubectl delete -f ingress-http-mul-url.yaml
```

### 4.2.3.2 单域名多URL支持子URL

范例: 单域名多URL, 支持子URL

```
#环境准备
[root@master1 ~]#kubectl create deployment pod-test1 --image=registry.cn-beijing.aliyuncs.com/wangxiaochun/pod-test:v0.1 --replicas=3
[root@master1 ~]#kubectl create service clusterip pod-test1 --tcp=80:80

[root@master1 ~]#kubectl create deployment pod-test2 --image=registry.cn-beijing.aliyuncs.com/wangxiaochun/pod-test:v0.2 --replicas=3
[root@master1 ~]#kubectl create service clusterip pod-test2 --tcp=80:80
```

#### #资源文件

```
[root@master1 ~]#cat ingress-http-mul-suburl.yml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$2 #正则表达式
  generation: 1
  name: ingress-http-mul-suburl
spec:
  ingressClassName: nginx
  rules:
    - host: www.wang.org
      http:
        paths:
          - backend:
              service:
                name: pod-test1
                port:
                  number: 80
              path: /v1(/|$)(.*) #正则表达式
              #pathType: Exact
              pathType: ImplementationSpecific #k8s-v1.32.0不再支持Exact,需要修改类型
          - backend:
              service:
                name: pod-test2
                port:
                  number: 80
              path: /v2(/|$)(.*) #正则表达式
              #pathType: Exact
              pathType: ImplementationSpecific #k8s-v1.32.0不再支持Exact,需要修改类型
```

#### #应用

```
[root@master1 ~]#kubectl apply -f ingress-http-mul-suburl.yml
```

#### #查看

```
[root@master1 ~]#kubectl describe ingress ingress-http-mul-suburl
```

Name: ingress-http-mul-suburl

Labels: <none>

Namespace: default

Address: 10.0.0.99

Ingress Class: nginx

Default backend: <default>

Rules:

Host	Path	Backends
----	----	-----

www.wang.org

/v1(/|\$)(.\*) pod-test1:80 (10.244.3.105:80)

/v2(/|\$)(.\*) pod-test2:80 (10.244.3.106:80)

Annotations: nginx.ingress.kubernetes.io/rewrite-target: /\$2

Events:

Type	Reason	Age	From	Message
----	-----	----	----	-----
Normal	Sync	6m24s (x2 over 6m57s)	nginx-ingress-controller	scheduled for sync

#集群外测试访问, 支持子URL

```
C:\Users\Wang>curl -H"host: www.wang.org" 10.0.0.99/v1
kubernetes pod-test v0.1!! ClientIP: 10.244.1.182, ServerName: pod-test1-
5f8f7698d8-p2pst, ServerIP: 10.244.3.105!

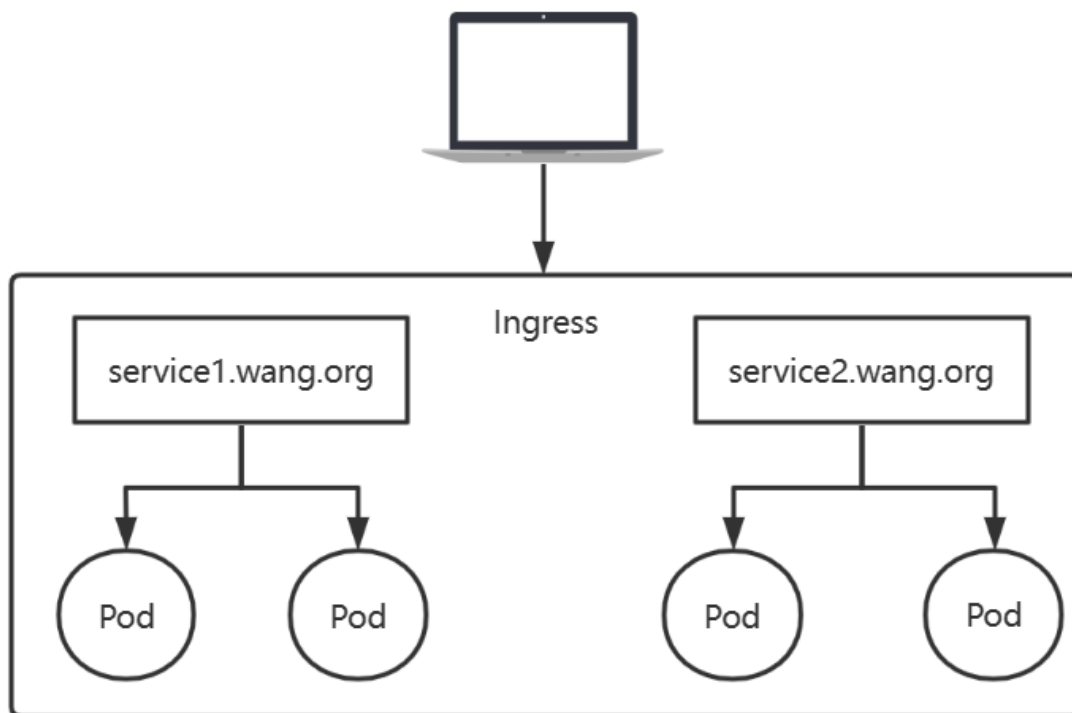
C:\Users\Wang>curl -H"host: www.wang.org" 10.0.0.99/v1/hostname
ServerName: pod-test1-5f8f7698d8-p2pst

C:\Users\Wang>curl -H"host: www.wang.org" 10.0.0.99/v2
kubernetes pod-test v0.2!! ClientIP: 10.244.1.182, ServerName: pod-test2-
f7d78c544-n25xn, ServerIP: 10.244.3.106!

C:\Users\Wang>curl -H"host: www.wang.org" 10.0.0.99/v2/hostname
ServerName: pod-test2-f7d78c544-n25xn

#清理环境
[root@master1 ~]#kubectl delete -f yaml/ingress-http-mul-suburl.yaml
```

## 4.2.4 多域名案例



Name based virtual hosting 基于虚拟主机名实现，为每个应用使用一个专有的主机名，并基于这些名称完成不同应用间的流量转发

- 每个FQDN对应于Ingress Controller上的一个虚拟主机的定义
- 同一组内的应用的流量，由Ingress Controller根据调度算法完成请求调度

范例: 多域名主机访问的环境准备两个HTTP应用

```
#访问 flask.wang.org/的时候，返回flask的结果
#访问 nginx.wang.org/的时候，返回nginx的结果
#如果前面的资源已删除，重新应用上面小节的资源文件生成deployment和对应的svc
[root@master1 ~]#kubectl apply -f ingress-deployment-svc.yaml

#再添加一个nginx的服务，定义资源文件
[root@master1 ~]#cat ingress-deployment-nginx.yaml
apiVersion: apps/v1
```



```

kind: Deployment
metadata:
  name: deployment-nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx-test
  template:
    metadata:
      labels:
        app: nginx-test
    spec:
      containers:
        - name: nginx-test
          image: registry.cn-beijing.aliyuncs.com/wangxiaochun/nginx:1.20.0
          imagePullPolicy: IfNotPresent
          ports:
            - containerPort: 80
              name: nginx
---
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx-test
  ports:
    - name: nginx
      port: 80
      targetPort: 80

```

#配置解析:为了避免多个service使用同一个后端端口,这里一定要为service的端口命名

#应用资源文件

```
[root@master1 ~]#kubectl apply -f ingress-deployment-nginx.yaml
```

#查看效果

```
[root@master1 ~]#kubectl get deployment,svc
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/deployment-nginx	3/3	3	3	88s
deployment.apps/deployment-test	3/3	3	3	73m

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
service/deployment-service	ClusterIP	10.99.228.134	<none>	80/TCP
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP
service/nginx-service	ClusterIP	10.99.61.212	<none>	80/TCP

范例: 多域名主机访问

#编辑Ingress资源定义文件

```
[root@master1 ~]#cat ingress-http-mul-host.yaml
```

```
apiVersion: networking.k8s.io/v1
```

```

kind: Ingress
metadata:
  name: ingress-mul-url
  annotations:
    #kubernetes.io/ingress.class: "nginx" #指定Ingress Controller的类型, k8s-
v1.32.0不再支持
    nginx.ingress.kubernetes.io/use-regex: "true" #指定后面rules定义的path使用的正则
表达式
    nginx.ingress.kubernetes.io/proxy-body-size: "100m" #客户端上传文件最大值, 默
认为1m
    nginx.ingress.kubernetes.io/proxy-connect-timeout: "60" #后端服务器的连接超时的时
间, 默认值为5s
    nginx.ingress.kubernetes.io/proxy-send-timeout: "120" #后端服务器数据回传超时时
间, 即规定时间之内后端服务器必须传完所有的数据, 默认值为60s
    nginx.ingress.kubernetes.io/proxy-read-timeout: "120" #后端服务器响应的超时时间,
默认值为60s
    #nginx.ingress.kubernetes.io/app-root: /index.html #指定默认页面文件
spec:
  ingressClassName: nginx #新版建议使用此项指定controller类型
  rules:
  - host: flask.wang.org
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: deployment-service
            port:
              name: http #匹配 servcie 中的端口 name: http
              #number: 80
  - host: nginx.wang.org
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: nginx-service
            port:
              name: nginx #匹配 servcie 中的端口 name: nginx

```

#配置解析: 这里面有两个主机访问的地址是同一个后端端口, 如果还用number的话, 就无法正常的识别导致同一个地址访问不同的后端效果

#应用资源文件

```
[root@master1 ~]#kubectl apply -f yaml/ingress-http-mul-host.yaml
```

#查看效果

```
[root@master1 ~]#kubectl get ingress
```

NAME	CLASS	HOSTS	ADDRESS
ingress-mul-url	<none>	flask.wang.org,nginx.wang.org	80

40s

```
[root@master1 ~]#kubectl describe ingress ingress-mul-url
```

```

Name:          ingress-mul-url
Namespace:     default

```

```

Address:          10.0.0.104
Default backend:  default-http-backend:80 (<error: endpoints "default-http-
backend" not found>)
Rules:
  Host                Path  Backends
  ----                -
  flask.wang.org      /    deployment-service:http
(10.244.3.66:80,10.244.3.67:80,10.244.5.38:80)
  nginx.wang.org      /    nginx-service:nginx
(10.244.3.68:80,10.244.4.51:80,10.244.5.39:80)
Annotations:      kubernetes.io/ingress.class: nginx
Events:
  Type    Reason    Age                From                Message
  ----    -
  Normal  Sync      17s (x2 over 65s)  nginx-ingress-controller  Scheduled for sync

#提前准备多个后端主机的域名
[root@master1 ~]#vim /etc/hosts
10.0.0.101 master1 master1.wang.org flask.wang.org nginx.wang.org

#访问效果
[root@master1 ~]#curl flask.wang.org
kubernetes pod-test v0.1!! ClientIP: 10.244.3.65, ServerName: deployment-test-
555f594785-rfhcz, ServerIP: 10.244.3.67!
[root@master1 ~]#curl nginx.wang.org
welcome to Nginx Web Site ! Nginx Version: 1.20.0

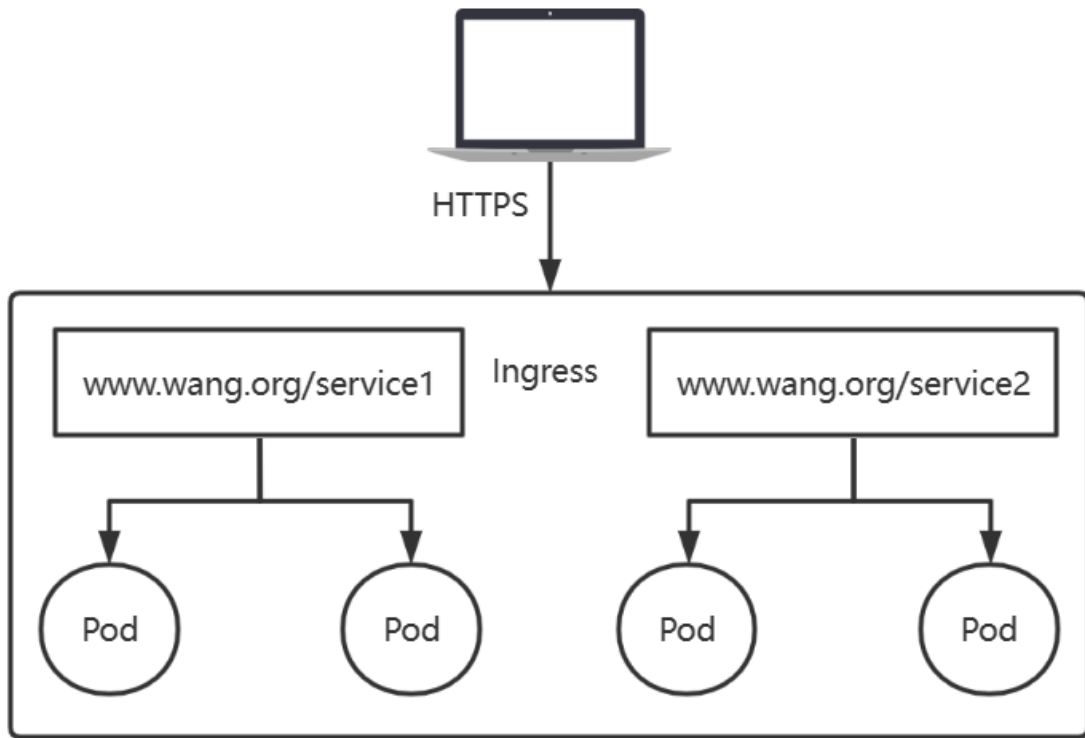
#再次修改hosts(配合前面hparoxy实现)
[root@master1 ~]#vim /etc/hosts
10.0.0.99 k8s k8s.wang.org www.wang.org flask.wang.org nginx.wang.org

[root@master1 ~]#curl nginx.wang.org
welcome to Nginx Web Site ! Nginx Version: 1.20.0
[root@master1 ~]#curl flask.wang.org
kubernetes pod-test v0.1!! ClientIP: 10.244.3.65, ServerName: deployment-test-
555f594785-l6v98, ServerIP: 10.244.3.66!

#清理环境
[root@master1 ~]#kubectl delete -f ingress-http-mul-host.yaml

```

## 4.2.5 HTTPS 案例



Ingress也可以提供TLS支持，但仅限于443/TCP端口

- 若TLS配置部分指定了不同的主机，则它们会根据通过SNI TLS扩展指定的主机名  
前提：Ingress控制器支持SNI在同一端口上复用
- TLS Secret必须包含名为tls.crt和的密钥tls.key，它们分别含有TLS的证书和私钥

#### 4.2.5.1 HTTPS 加密案例

准备对基础的flask应用进行https方式来进行访问

利用ingress的tls规则就可以帮助我们实现对http应用进行https安全加固

范例：实现HTTP自动跳转至HTTPS

```
#生成证书
[root@master1 ~]#mkdir tls && cd tls
[root@master1 tls]#(umask 077; openssl genrsa -out tls.key 2048)

#生成自签名证书，文件名可自定义
[root@master1 tls]#openssl req -new -x509 -key www.wang.org.key -out
www.wang.org.crt -subj "/CN=www.wang.org" -days 365

#创建对应的secret资源
[root@master1 tls]#kubectl create secret tls ingress-tls --
cert=./www.wang.org.crt --key=./www.wang.org.key

[root@master1 ~]#kubectl describe secrets ingress-tls
Name:          ingress-tls
Namespace:     default
Labels:        <none>
Annotations:   <none>

Type:          kubernetes.io/tls
```

Data

====

tls.crt: 1131 bytes

tls.key: 1679 bytes

#定义资源配置文件,实现HTTP自动跳转至HTTPS

[root@master1 ~]#cat ingress-http-tls-test.yaml

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: ingress-test

#annotations:

#kubernetes.io/ingress.class: "nginx"

spec:

ingressClassName: nginx #新版建议使用此项指定controller类型

rules:

- host: www.wang.org

http:

paths:

- path: /

pathType: Prefix

backend:

service:

name: deployment-service

port:

number: 80

#- hosts: m.wang.org

.....

# https 证书配置

tls:

- hosts:

- www.wang.org

secretName: ingress-tls

#- hosts: #多个域名分别对应不同的证书

# - m.wang.org

# secretName: ingress-tls-m

#应用资源文件

[root@master1 ~]#kubectl apply -f ingress-http-tls-test.yaml

#查看效果

[root@master1 ~]#kubectl get ingress

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
ingress-mul-url	<none>	www.wang.org	10.0.0.104	80	25m

[root@master1 ~]#kubectl describe ingress ingress-test

Name: ingress-test

Labels: <none>

Namespace: default

Address:

Ingress Class: <none>

Default backend: <default>

TLS:

ingress-tls terminates www.wang.org

Rules:

Host	Path	Backends
------	------	----------

----

----

-----

```

www.wang.org
/ deployment-service:80
(10.244.3.66:80,10.244.3.67:80,10.244.5.38:80)
Annotations:      kubernetes.io/ingress.class: nginx
Events:
  Type    Reason  Age                From              Message
  ----    -
  Normal  Sync    29s (x2 over 67s)  nginx-ingress-controller  Scheduled for sync

[root@master1 ~]#vim /etc/hosts
10.0.0.101 master1 master1.wang.org www.wang.org

#测试效果
[root@master1 ~]#curl www.wang.org
<html>
<head><title>308 Permanent Redirect</title></head>
<body>
<center><h1>308 Permanent Redirect</h1></center>
<hr><center>nginx</center>
</body>
</html>

#结果显示：所有访问http://www.wang.org的请求重定向HTTPS

[root@master1 ~]#curl -kL http://www.wang.org
kubernetes pod-test v0.1!! ClientIP: 10.244.3.65, ServerName: deployment-test-555f594785-rfhcz, ServerIP: 10.244.3.67!

[root@master1 ~]#curl -k https://www.wang.org
kubernetes pod-test v0.1!! ClientIP: 10.244.3.65, ServerName: deployment-test-555f594785-rfhcz, ServerIP: 10.244.3.67!
[root@master1 ~]#curl -k https://www.wang.org
kubernetes pod-test v0.1!! ClientIP: 10.244.3.65, ServerName: deployment-test-555f594785-9mmbq, ServerIP: 10.244.5.38!
[root@master1 ~]#curl -k https://www.wang.org
kubernetes pod-test v0.1!! ClientIP: 10.244.3.65, ServerName: deployment-test-555f594785-l6v98, ServerIP: 10.244.3.66!

#查看证书有效期
#注意：openssl命令的版本过低将无法看到过期时间，比如：CentOS8
[root@ubuntu2204 ~]#openssl s_client www.wang.org:443
CONNECTED(00000003)
depth=0 C = CN, ST = Beijing, L = Beijing, O = SRE, CN = www.wang.org
verify error:num=18:self-signed certificate
verify return:1
depth=0 C = CN, ST = Beijing, L = Beijing, O = SRE, CN = www.wang.org
verify return:1
---
Certificate chain
 0 s:C = CN, ST = Beijing, L = Beijing, O = SRE, CN = www.wang.org
  i:C = CN, ST = Beijing, L = Beijing, O = SRE, CN = www.wang.org
  a:PKEY: rsaEncryption, 2048 (bit); sigalg: RSA-SHA256
  v:NotBefore: Jan  9 04:01:09 2025 GMT; NotAfter: Jan  7 04:01:09 2035 GMT

#清理环境

```

```
[root@master1 ~]#kubectl delete -f yaml/ingress-http-tls-test.yaml
```

### 4.2.5.2 Kubernetes dashboard 实现 Https

Kubernetes dashboard 是 Kubernetes的基于web的图形化管理平台

基于安全原因,应该要求使用https方式来进行访问Kubernetes的dashboard。

可以通过service的方式实现正常的请求访问。比如 <https://10.0.0.101:30000/#/login>

有了ingress, 也可以通过域名更加方便的访问dashboard了。

除了我们之前所说的动态url的方式之外, 安全原因还需要tls认证。

关于对集群内部的ingress的tls访问方式, 可以借助于ingress controller的annotation的方式向nginx中传递一些属性, 从而上nginx支持相应的功能。

实现方式上与我们上一节的方式有所不同

通过两个annotation中的规则

- ingress.kubernetes.io/ssl-passthrough: "true"  
因为后面的dashboard本身就做了tls功能, 所以这里进行TLS透传, 即实现四层代理功能
- nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"  
指定对后端启用https协议的访问。

参考资料: <https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/annotation/>

#### 4.2.5.2.1 环境准备提前部署kubernetes的dashboard

范例: 环境准备, 提前部署kubernetes的dashboard

```
#部署metrics-server, 可选
[root@master1 ~]#wget https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml

#修改镜像
[root@master1 ~]#vim components.yaml
- --kubelet-insecure-tls #添加此行
  image: registry.k8s.io/metrics-server/metrics-server:v0.7.2
  image: registry.cn-hangzhou.aliyuncs.com/google_containers/metrics-server:v0.7.2

[root@master1 ~]#grep kind: components.yaml
kind: ServiceAccount
kind: ClusterRole
kind: ClusterRole
kind: RoleBinding
  kind: Role
- kind: ServiceAccount
kind: ClusterRoleBinding
  kind: ClusterRole
- kind: ServiceAccount
kind: ClusterRoleBinding
  kind: ClusterRole
- kind: ServiceAccount
kind: Service
kind: Deployment
```

```
kind: APIService
[root@master1 ~]#kubectl apply -f components.yaml
serviceaccount/metrics-server created
clusterrole.rbac.authorization.k8s.io/system:aggregated-metrics-reader created
clusterrole.rbac.authorization.k8s.io/system:metrics-server created
rolebinding.rbac.authorization.k8s.io/metrics-server-auth-reader created
clusterrolebinding.rbac.authorization.k8s.io/metrics-server:system:auth-delegator
created
clusterrolebinding.rbac.authorization.k8s.io/system:metrics-server created
service/metrics-server created
deployment.apps/metrics-server created
apiservice.apiregistration.k8s.io/v1beta1.metrics.k8s.io created
```

#下载并修改配置文件

#<https://github.com/kubernetes/dashboard>

```
[root@master1 ~]#wget
https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0/aio/deploy/recomm
ended.yaml
[root@master1 ~]#vim recommended.yaml
```

```
.....
kind: Service
apiVersion: v1
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kubernetes-dashboard
spec:
  type: LoadBalancer | NodePort          # 新增此行, 指定类型
  ports:
    - port: 443
      targetPort: 8443
      nodePort: 30000          #新增此行, 指定该端口作为外界访问入口的端口, 端口必须在30000-
32767之间
```

```
.....
  containers:
    - name: kubernetes-dashboard
      image: registry.cn-beijing.aliyuncs.com/wangxiaochun/dashboard:v2.0.0
      #image: kubernetesui/dashboard:v2.0.0
      #image: harbor.wang.org/kubernetes/dashboard:v2.0.0 #可选, 修改此行为
harbor镜像地址
```

```
.....
  containers:
    - name: dashboard-metrics-scraper
      image: registry.cn-beijing.aliyuncs.com/wangxiaochun/metrics-
scraper:v1.0.4
      #image: kubernetesui/metrics-scraper:v1.0.4
      #image: harbor.wang.org/kubernetes/metrics-scraper:v1.0.4 #可选, 修改此行
harbor镜像地址
```

#应用配置,会自动下载两个镜像dashboard和metrics-scraper到某个node节点

```
[root@master1 ~]#kubectl apply -f recommended.yaml
namespace/kubernetes-dashboard created
serviceaccount/kubernetes-dashboard created
service/kubernetes-dashboard created
secret/kubernetes-dashboard-certs created
secret/kubernetes-dashboard-csrf created
```



```
secret/kubernetes-dashboard-key-holder created
configmap/kubernetes-dashboard-settings created
role.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrole.rbac.authorization.k8s.io/kubernetes-dashboard created
rolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
deployment.apps/kubernetes-dashboard created
service/dashboard-metrics-scraper created
deployment.apps/dashboard-metrics-scraper created
```

#确认Pod运行正常

```
[root@master1 ~]#kubectl get all -n kubernetes-dashboard
```

NAME	READY	STATUS	RESTARTS
AGE			
pod/dashboard-metrics-scraper-7448c88bd7-q5wmz	1/1	Running	0
34m			
pod/kubernetes-dashboard-5cf5468895-bxzn2	1/1	Running	0
34m			

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP
PORT(S) AGE			
service/dashboard-metrics-scraper	ClusterIP	10.110.79.154	<none>
8000/TCP 34m			
service/kubernetes-dashboard	LoadBalancer	10.100.122.245	10.0.0.11
443:30000/TCP 34m			

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/dashboard-metrics-scraper	1/1	1	1	34m
deployment.apps/kubernetes-dashboard	1/1	1	1	34m

NAME	DESIRED	CURRENT	READY
AGE			
replicaset.apps/dashboard-metrics-scraper-7448c88bd7	1	1	1
34m			
replicaset.apps/kubernetes-dashboard-5cf5468895	1	1	1
34m			

#生成登录需要的token

```
[root@master1 ~]#kubectl create serviceaccount dashboard-admin -n kube-system
```

```
[root@master1 ~]#kubectl create clusterrolebinding dashboard-admin --
clusterrole=cluster-admin --serviceaccount=kube-system:dashboard-admin
```

```
[root@master1 ~]#cat security-dashboard-admin-secret.yaml
```

```
apiVersion: v1
kind: Secret
type: kubernetes.io/service-account-token
metadata:
  name: dashboard-admin-secret
  namespace: kube-system
  annotations:
    kubernetes.io/service-account.name: "dashboard-admin"
```

```
[root@master1 ~]#kubectl apply -f security-dashboard-admin-secret.yaml
```

#查看token

```
[root@master1 ~]#kubectl describe secrets -n kube-system dashboard-admin-secret
```

Name: dashboard-admin-secret  
Namespace: kube-system  
Labels: <none>  
Annotations: kubernetes.io/service-account.name: dashboard-admin  
kubernetes.io/service-account.uid: 6dbd0b89-7eee-4d45-af3b-ab5a50d6456a

Type: kubernetes.io/service-account-token

Data

=====

token: #复制下面的内容

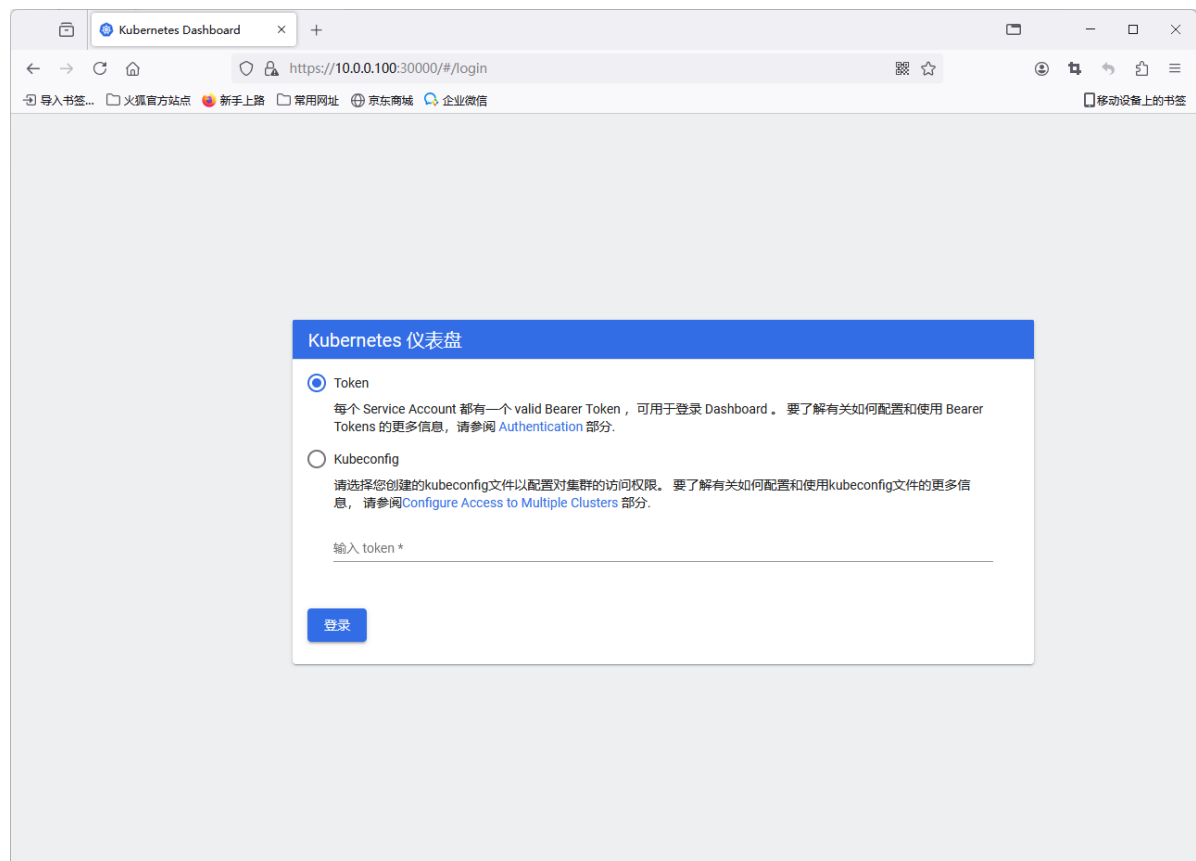
eyJhbGciOiJSUzI1NiIsImtpZCI6Imp2aFBoSUXLVU5LUUC2aU1Lew1oaH1IbjVWZXBuZD1VN2tSRnhoZ0N1RHMifQ.eyJpc3MiOiJrdWJ1cm5ldGVL3N1cnZpY2VhY2NvdW50Iiwia3ViZXJ1cy5pbY9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJrdWJ1LXN5c3R1bSIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJkYXNoYm9hcmQtYWRtaW4tc2VjcmV0Iiwia3ViZXJ1cy5pbY9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWJ1bnQubmFtZSI6ImRhc2hib2FyZC1hZG1pb1IsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VydmljZS1hY2NvdW50LnVpZCI6IjZkYmQwYjg5LTdlZm50LWU0NS1hZjNiLWFiNWElMGQ2NDU2YSIsInN1YiI6InN5c3R1bTpkZXJ2aWNlYWNjb3VudDprdwJ1LXN5c3R1bTpkYXNoYm9hcmQtYWRtaW4ifQ.DgxeIpL1hRLYoHAcck32h0Iyaen2xGuRpbFvscy1vgJ7yOHW60Bnt3DL--

NYRY1R8wseH1nYesvyr0ZRI7lh1D6ShAtWL3jNkrE2uTJaQRXj9\_9RD4UZWN4SJ9OhJW4otfx-u-zR2kF5pIXgY3gaU5rMXjLxmesPEXBxNMAZOBl27a6BgXLCR\_5LxQVFI1bIq7paI0Jn8CYNHLUAmImt5BqziCRo46mvUabLKsmrwqa1UCS8hdpZAe5F-m5cFiWQtIoNo5dGHSdxSCwRfLuL7Sy19gsdwJ34F75KzsAzxXH97vtn1XGGFp7\_rP\_CD4x0GLthtDmC yNJsnfacDgXoA

ca.crt: 1107 bytes

namespace: 11 bytes

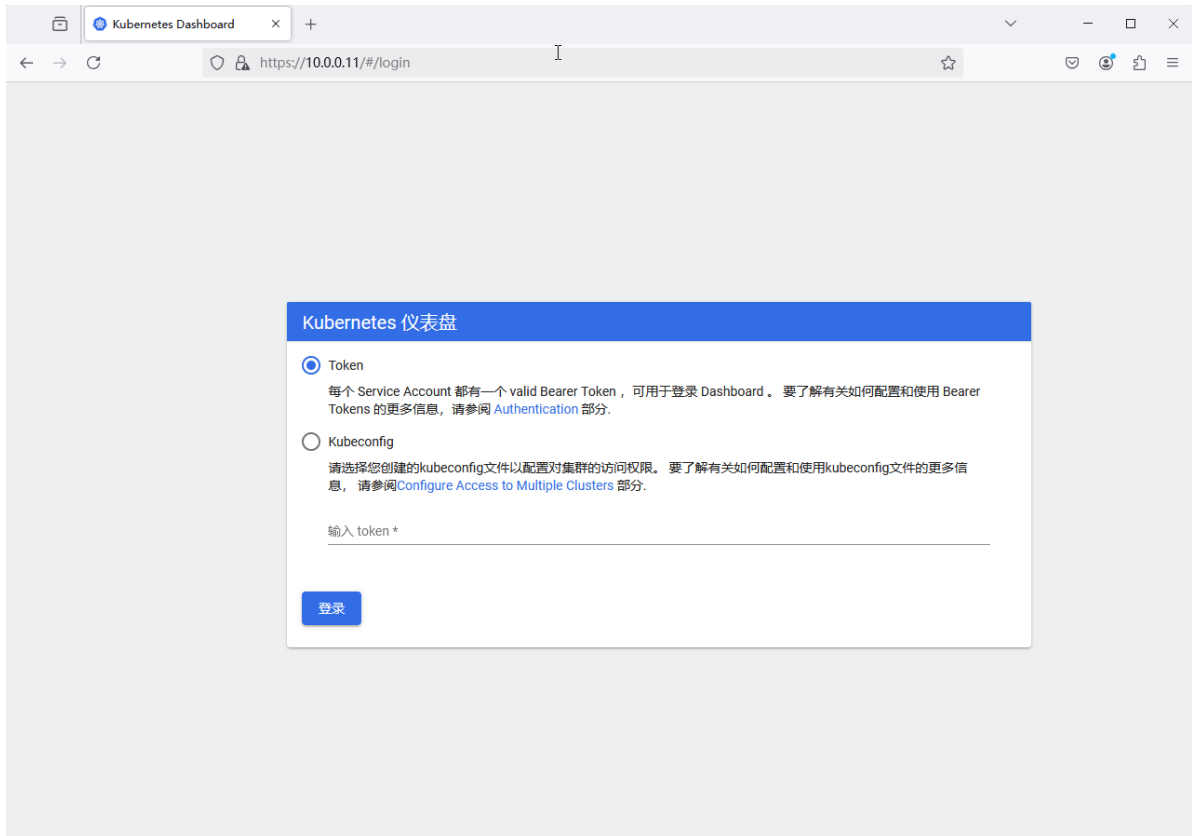
因为证书是自签名，Chrome和Edge不能信任，无法正常打开，可以换firefox浏览器访问



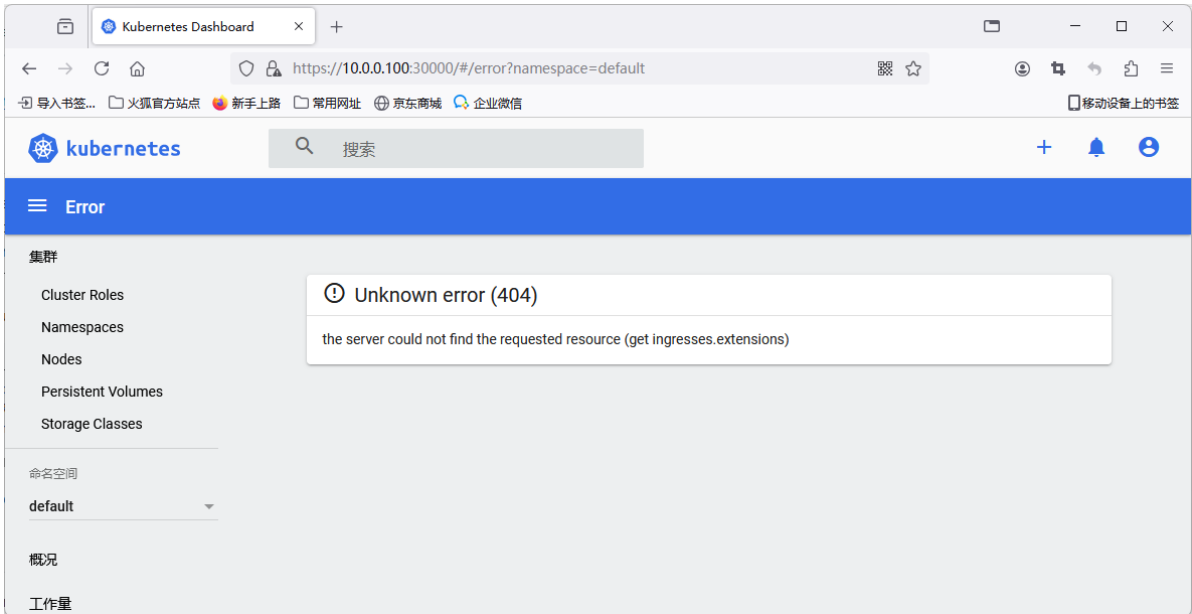
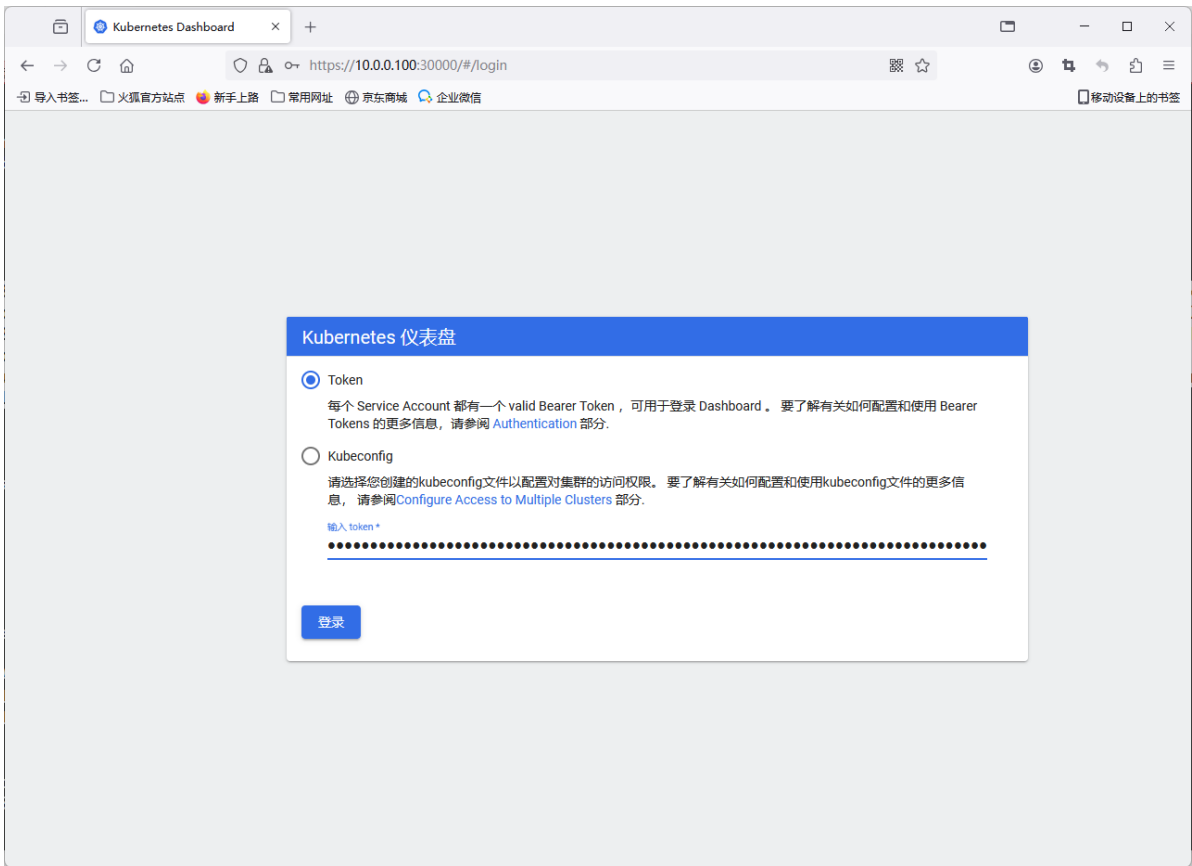
如何配置loadbalancer 的SVC，也可以通过EXTERNAL-IP访问

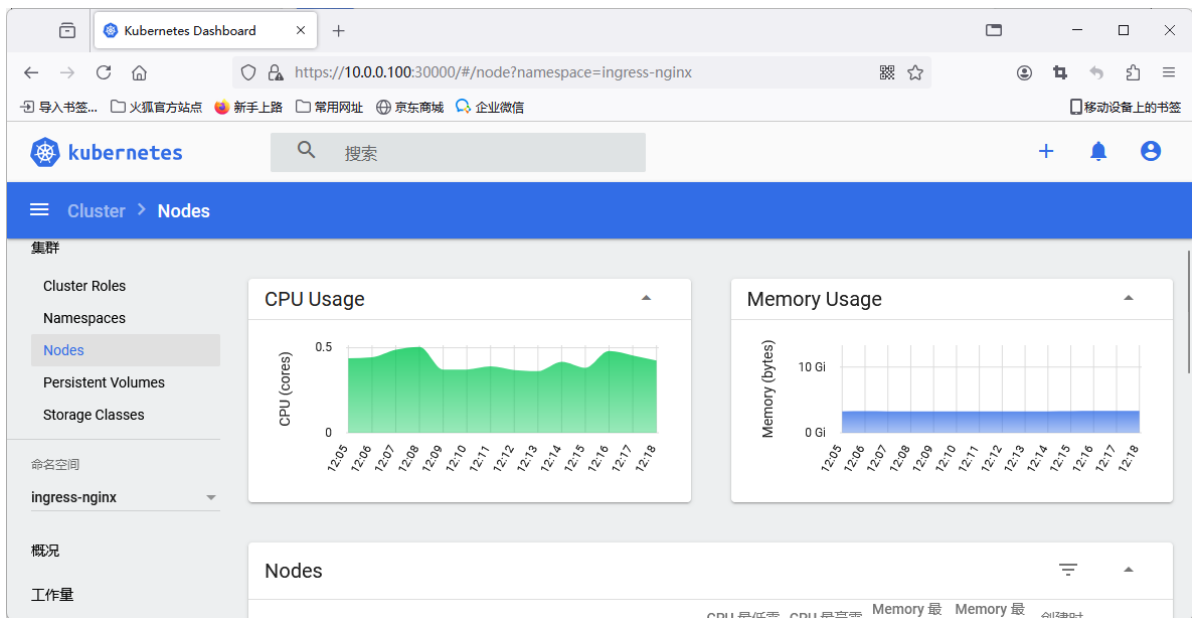
```
[root@master1 ingress]#kubectl get svc -n kubernetes-dashboard
```

NAME	AGE	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
dashboard-metrics-scraper	12m	ClusterIP	10.109.42.122	<none>	8000/TCP
kubernetes-dashboard	12m	LoadBalancer	10.109.255.200	10.0.0.11	443:40000/TCP



#复制生成的Token登录





#### 4.2.5.2.2 通过 Ingress 实现 kubernetes dashboard 的访问

范例: 通过 Ingress 实现 kubernetes dashboard 的访问

```
#定制ingress资源配置文件
[root@master1 ~]#cat ingress-https-dashboard.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-dashboard
  namespace: kubernetes-dashboard
  annotations:
    #kubernetes.io/ingress.class: "nginx"                #旧版用法
    ingress.kubernetes.io/ssl-passthrough: "true"      #进行TLS透传，四层代理
    nginx.ingress.kubernetes.io/backend-protocol: "HTTPS" #指定对后端启用https协议的访问
    nginx.ingress.kubernetes.io/rewrite-target: /$2

spec:
  ingressClassName: "nginx"                             #新版用法
  rules:
  - host: www.wang.org
    http:
      paths:
      - path: /dashboard(/|$(.*)
        #pathType: Prefix
        pathType: ImplementationSpecific #k8s-v1.32.0需要修改类型
        backend:
          service:
            name: kubernetes-dashboard
            port:
              number: 443

#- host: dashboard.wang.org #下面相关配置需要配合
nginx.ingress.kubernetes.io/rewrite-target: /$1 使用
#http:
#  paths:
#    - path: /(.*)
#      pathType: Prefix
#      backend:
```

```
#      service:
#      name: kubernetes-dashboard
#      port:
#      number: 443
```

#应用资源文件

```
[root@master1 ~]#kubectl apply -f ingress-https-dashboard.yaml
```

#查看效果

```
[root@master1 ~]#kubectl get ingress -n kubernetes-dashboard
```

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
ingress-dashboard	<none>	www.wang.org		80	16s

```
[root@master1 ~]#kubectl describe ingress -n kubernetes-dashboard
```

```
Name:          ingress-dashboard
Namespace:     kubernetes-dashboard
Address:       10.0.0.104
Default backend: default-http-backend:80 (<error: endpoints "default-http-backend" not found>)
```

Rules:

Host	Path	Backends
www.wang.org	/dashboard(/ \$)(.*)	kubernetes-dashboard:443

(10.244.4.34:8443)

```
Annotations:  ingress.kubernetes.io/ssl-passthrough: true
              kubernetes.io/ingress.class: nginx
              nginx.ingress.kubernetes.io/backend-protocol: HTTPS
              nginx.ingress.kubernetes.io/rewrite-target: /$2
```

Events:

Type	Reason	Age	From	Message
Normal	Sync	56s (x2 over 80s)	nginx-ingress-controller	Scheduled for sync

#在客户端完成发下名称解析

```
10.0.0.101 master1 master1.wang.org www.wang.org
```

#HTTP自动跳转HTTPS

```
[root@ubuntu2004 ~]#curl http://www.wang.org/dashboard
```

```
<html>
<head><title>308 Permanent Redirect</title></head>
<body>
<center><h1>308 Permanent Redirect</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

```
[root@ubuntu2004 ~]#curl -Lk http://www.wang.org/dashboard
```

```
<!--
```

Copyright 2017 The Kubernetes Authors.

Licensed under the Apache License, Version 2.0 (the "License");  
you may not use this file except in compliance with the License.  
You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

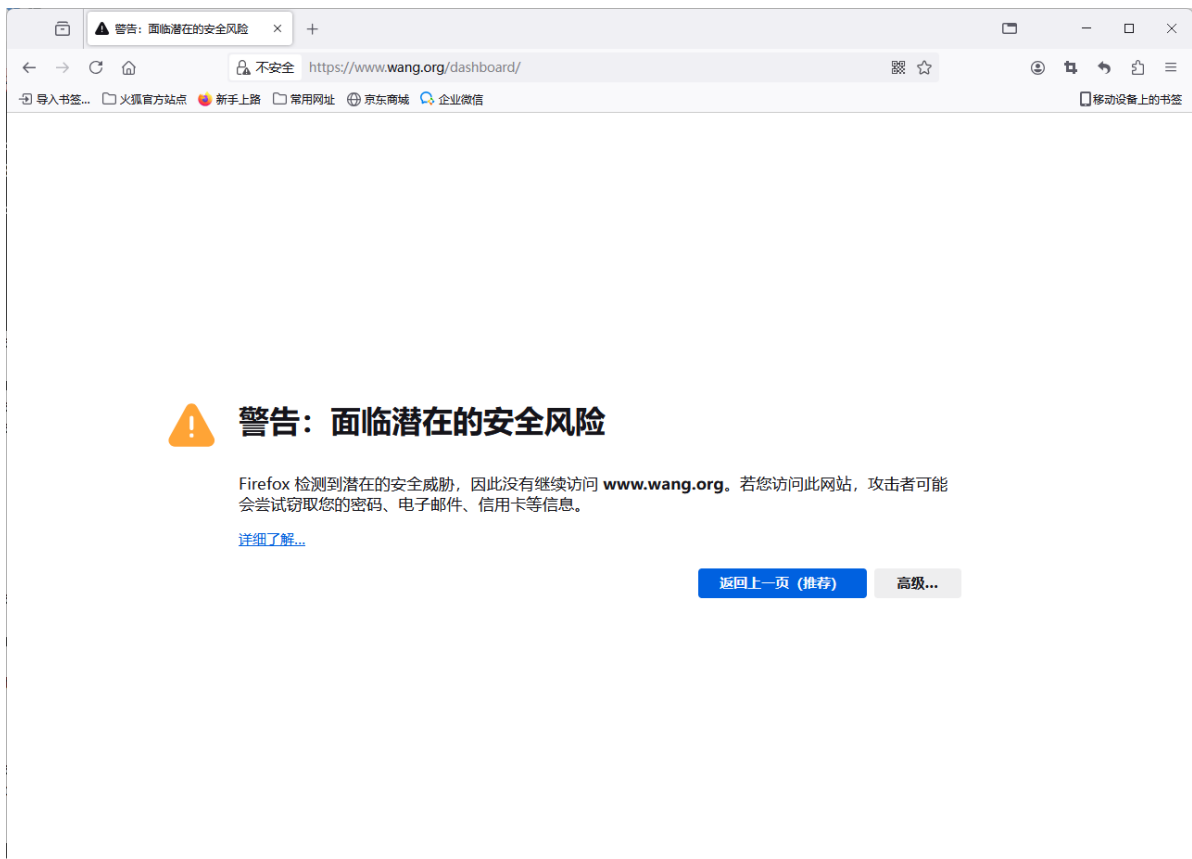
Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

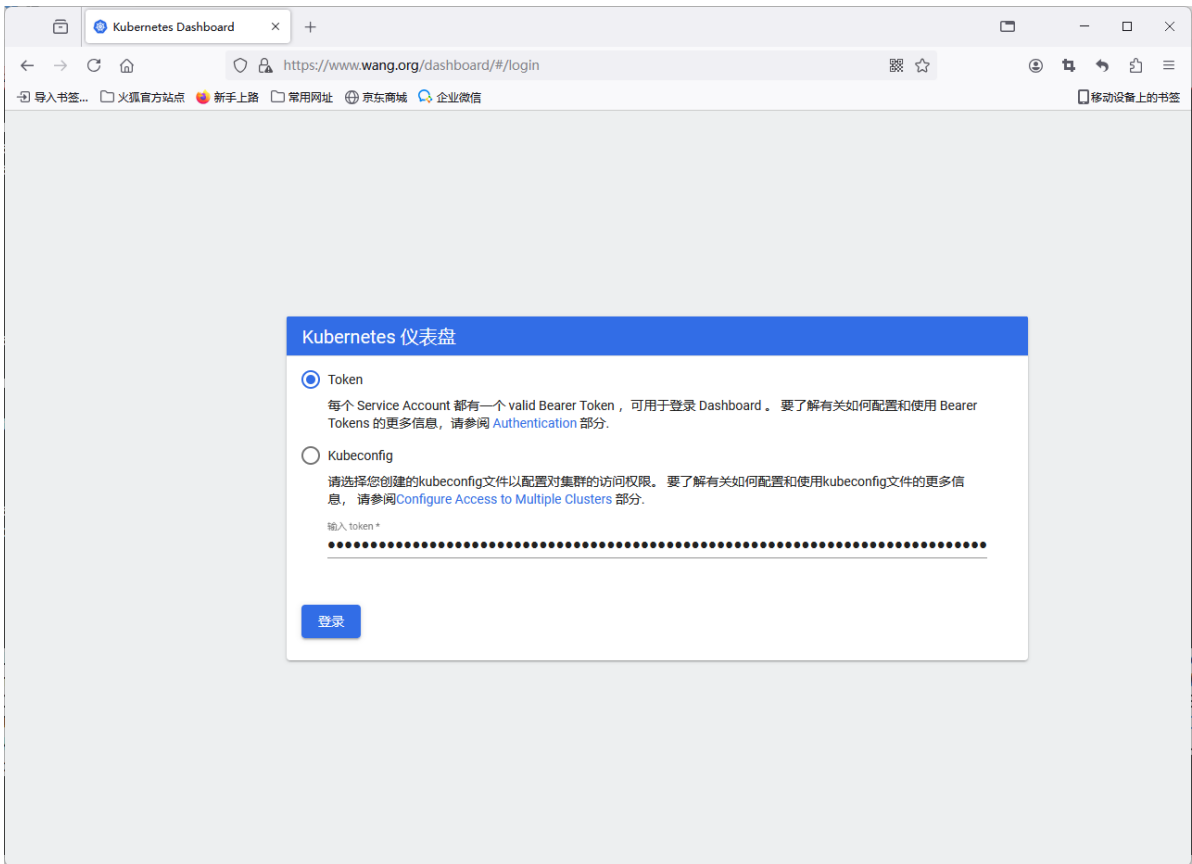
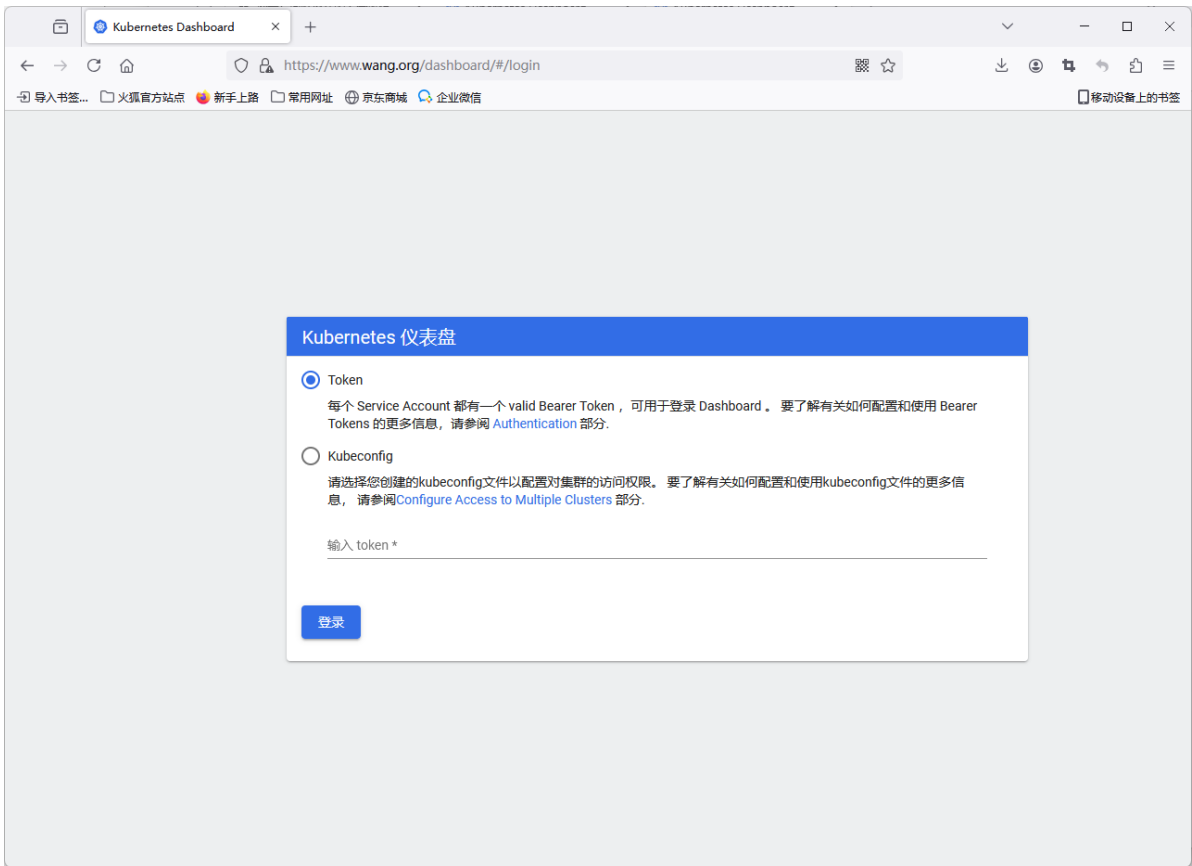
```
--><!DOCTYPE html><html lang="en" dir="ltr"><head>
  <meta charset="utf-8">
  <title>Kubernetes Dashboard</title>
  <link rel="icon" type="image/png" href="assets/images/kubernetes-logo.png">
  <meta name="viewport" content="width=device-width">
<style>html,body{height:100%;margin:0}*::-webkit-
scrollbar{background:transparent;height:8px;width:8px}</style><link
rel="stylesheet" href="styles.243e6d874431c8e8.css" media="print"
onload="this.media='all'"><noscript><link rel="stylesheet"
href="styles.243e6d874431c8e8.css"></noscript></head>

<body>
  <kd-root></kd-root>
<script src="runtime.134ad7745384bed8.js" type="module"></script><script
src="polyfills.5c84b93f78682d4f.js" type="module"></script><script
src="scripts.2c4f58d7c579cacb.js" defer></script><script
src="en.main.3550e3edca7d0ed8.js" type="module"></script>

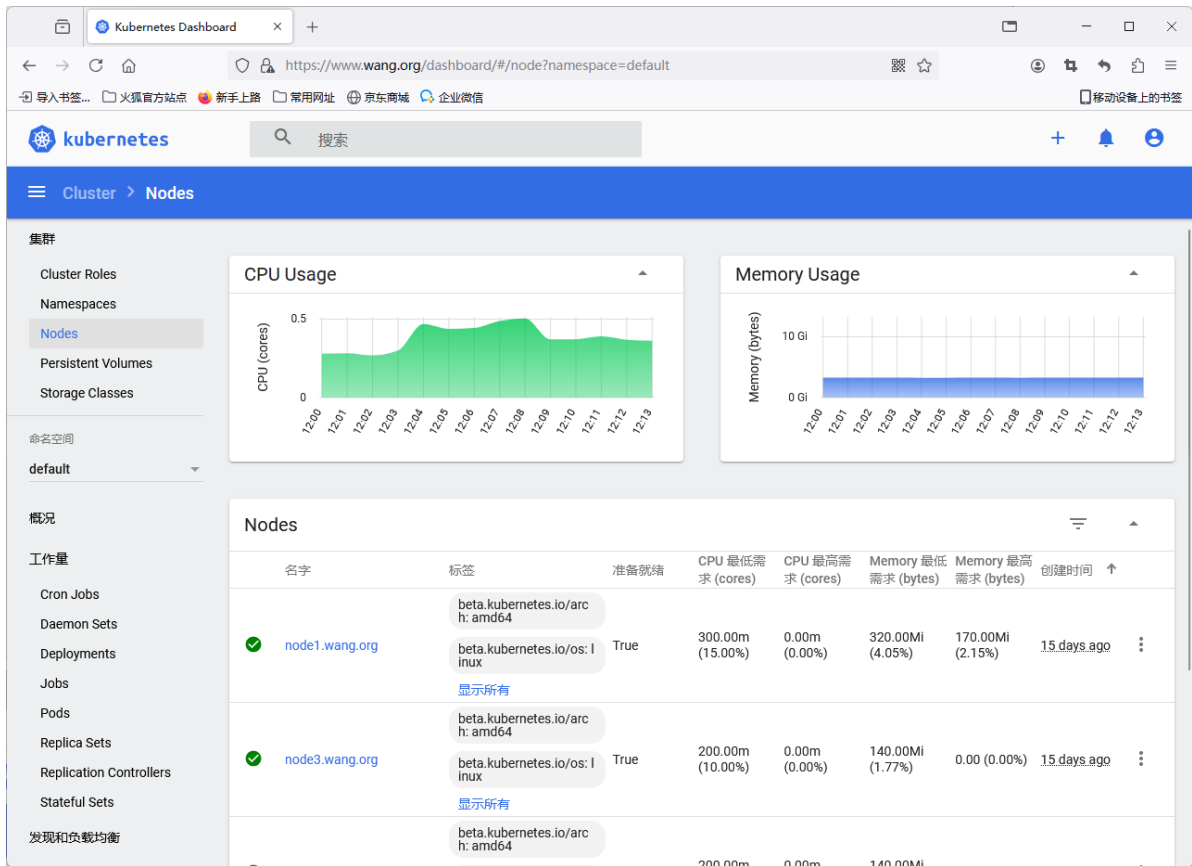
</body></html>
```

#浏览器访问 <https://www.wang.org/dashboard/> 查看效果，注意，dashboard后面有/  
#结果显示：借助于nginx-controller的tls功能可以正常的实现相关的https的效果。









## 5 Ingress Nginx 实现蓝绿 BlueGreen 和灰度Canary 发布

### 5.1 Ingress Nginx 进行BlueGreen 和 Canary 灰度发布说明

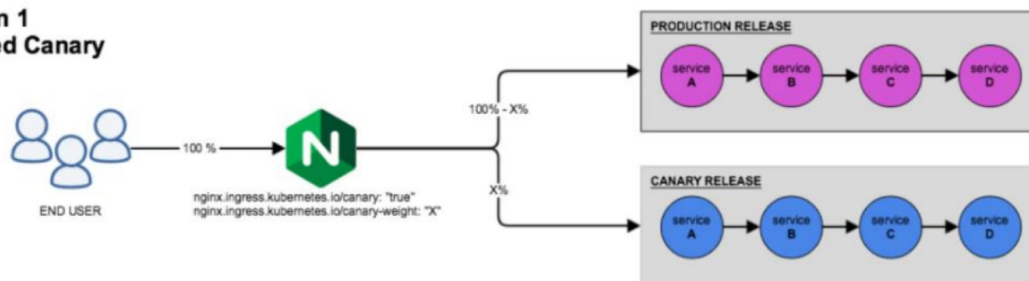
Service 虽然支持流量分配,但是只支持基于Pod的数量或比例实现,而不支持基于Header,cookie,权重等更为清晰的流量发配策略

Ingress-Nginx支持配置Ingress Annotations来实现不同场景下的灰度发布和测试,它能够满足金丝雀发布、蓝绿部署与A/B测试等不同的业务场景

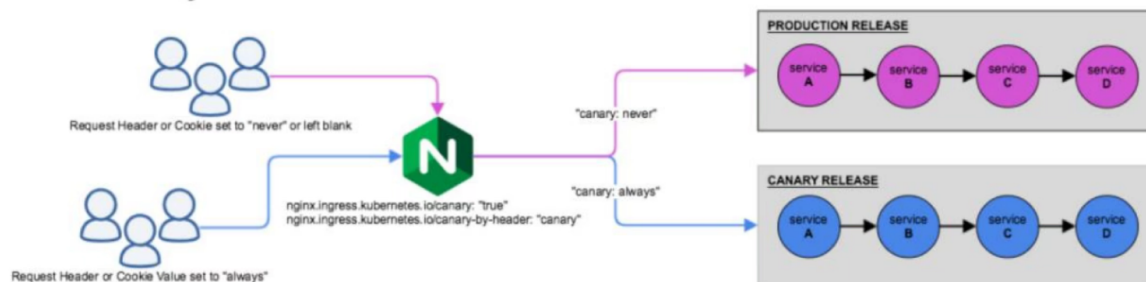
注意: Ingress-Nginx 只能支持南北向的流量发布,而东西向流量的发布可以利用工作负载型如 deployment的更新策略或者服务网格技术实现

**Ingress Nginx的流量发布机制:**

### Option 1 Weight-Based Canary



### Option 2 User-Based Canary



- 蓝绿：
  - production: 100%, canary: 0%
  - production: 0%, canary: 100% --> Canary变成后面的Production
- 金丝雀 Canary:
  - 流量比例化切分: 逐渐调整
  - 流量识别，将特定的流量分发给Canary:
    - By-Header: 基于特定的标头识别
      - Header 值默认: 只有Always 或 Nerver 两种值
      - Header 值自定义
      - Header 值可以基于正则表达式Pattern进行匹配
    - By-Cookie: 基于Cookie识别

### 基于Ingress Nginx的Canary规则

Ingress Nginx 的 Annotations支持的Canary规则，Annotations 和 Label 相似也是保存资源对象上的元数据，但不能被标签选择器选择，且没有Label的名称最长63个字符的限制

- nginx.ingress.kubernetes.io/canary-weight:
  - 基于服务权重进行流量切分，适用于蓝绿或灰度发布，权重范围0 - 100按百分比将请求路由到 Canary Ingress中指定的服务
  - 权重为 0 意味着该金丝雀规则不会向Canary入口的服务发送任何请求
  - 权重为100意味着所有请求都将被发送到 Canary 入口
- nginx.ingress.kubernetes.io/canary-by-cookie:
  - 基于 cookie 的流量切分，适用于灰度发布与 A/B 测试
  - cookie 的值设置为 always 时，它将被路由到Canary入口
  - cookie 的值设置为 never 时，请求不会被发送到Canary入口

对于任何其他值，将忽略 cookie 并将请求与其他金丝雀规则进行优先级的比较，默认转发给旧版本

- `nginx.ingress.kubernetes.io/canary-by-header`:

基于该Annotation中指定Request Header进行流量切分，适用于灰度发布以及A/B测试

在请求报文中，若存在该Header且其值为always时，请求将会被发送到Canary版本，注意：always是大小敏感的

若存在该Header且其值为never，请求将不会被发送至Canary版本

若存在该Header且其值为任何其它值，将忽略该Annotation指定的Header，并按优先级将请求与其他金丝雀规则进行比较，默认转发给旧版本

若不存在该Header时，请求将不会被发送至Canary版本

- `nginx.ingress.kubernetes.io/canary-by-header-value`:

基于该Annotation中指定的Request Header的值进行流量切分，Header名称则由前一个Annotation (`nginx.ingress.kubernetes.io/canary-by-header`) 进行指定

请求报文中存在指定的Header，且其值与该Annotation的值匹配时，它将被路由到Canary版本

对于任何其它值，将忽略该Annotation，默认转发给旧版本

- `nginx.ingress.kubernetes.io/canary-by-header-pattern`

同`canary-by-header-value`的功能类似，但该Annotation基于正则表达式匹配Request Header的值

若该Annotation与`canary-by-header-value`同时存在，则该Annotation会被忽略，默认转发给旧版本

### 规则的应用次序

- Canary规则会按特定的次序进行评估
- 优先级从低到高顺序：`canary-weight` -> `canary-by-cookie` -> `canary-by-header`

## 5.2 实战案例

### 5.2.1 范例：初始环境准备新旧两个版本应用和Ingress

```
#准备新旧版本对应的各自独立的两套deployment和服务
```

```
[root@master1 ~]#cat deploy-pod-test-v1.yaml
```

```
apiVersion: apps/v1
```

```
kind: Deployment
```

```
metadata:
```

```
  labels:
```

```
    app: pod-test
```

```
  name: pod-test-v1
```

```
spec:
```

```
  replicas: 1
```

```
  selector:
```

```
    matchLabels:
```

```
      app: pod-test
```

```
      version: v0.1
```

```
  strategy: {}
```

```
  template:
```

```
    metadata:
```

```
      labels:
```

```
        app: pod-test
```

```
        version: v0.1
```

```
spec:
  containers:
  - image: registry.cn-beijing.aliyuncs.com/wangxiaochun/pod-test:v0.1
    name: pod-test
---
apiVersion: v1
kind: Service
metadata:
  labels:
    app: pod-test
  name: pod-test-v1
spec:
  ports:
  - name: http-80
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: pod-test
    version: v0.1
  type: ClusterIP
```

[root@master1 ~]#cat deploy-pod-test-v2.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: pod-test
  name: pod-test-v2
spec:
  replicas: 1
  selector:
    matchLabels:
      app: pod-test
      version: v0.2
  strategy: {}
  template:
    metadata:
      labels:
        app: pod-test
        version: v0.2
    spec:
      containers:
      - image: registry.cn-beijing.aliyuncs.com/wangxiaochun/pod-test:v0.2
        name: pod-test
---
apiVersion: v1
kind: Service
metadata:
  labels:
    app: pod-test
  name: pod-test-v2
spec:
  ports:
  - name: http-80
    port: 80
```

```
protocol: TCP
targetPort: 80
selector:
  app: pod-test
  version: v0.2
type: ClusterIP
```

#部署新旧两个版本

```
[root@master1 ~]#kubectl apply -f deploy-pod-test-v1.yaml -f deploy-pod-test-v2.yaml
```

```
[root@master1 ~]#kubectl get svc
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	28d
pod-test-v1	ClusterIP	10.111.30.185	<none>	80/TCP	78s
pod-test-v2	ClusterIP	10.107.142.163	<none>	80/TCP	75s

```
[root@master1 ~]#kubectl get deployments.apps
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
pod-test-v1	1/1	1	1	95s
pod-test-v2	1/1	1	1	92s

#创建Ingress对应旧版本的应用

```
[root@master1 ~]#cat ingress-pod-test.yaml
```

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: pod-test
  #annotations: ingressClassName
  # kubernetes.io/ingress.class: nginx
spec:
  ingressClassName: nginx
  rules:
  - host: www.wang.org
    http:
      paths:
      - backend:
          service:
            name: pod-test-v1
            port:
              number: 80
        path: /
        pathType: Prefix
```

#应用

```
[root@master1 ~]#kubectl apply -f ingress-pod-test.yaml
```

#查看

```
[root@master1 ~]#kubectl get ingress
```

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
pod-test	<none>	www.wang.org		80	9s

#集群外客户端访问

```
[root@ubuntu2004 ~]#curl -H"host:www.wang.org" http://10.0.0.99/
kubernetes pod-test v0.1!! ClientIP: 10.244.2.37, ServerName: pod-test-v1-6ddc47c8cd-1mcmh, ServerIP: 10.244.2.38!
```

#持续执行观察

```
[root@ubuntu2004 ~]#while true;do curl -k -H"host:www.wang.org"
https://10.0.0.99/;sleep 0.5;done
```

## 5.2.2 范例：蓝绿发布

#修改上前的Ingress清单文件,对应使用新版本的应用

```
[root@master1 ~]#vim ingress-pod-test.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: pod-test
  #annotations:
  #  kubernetes.io/ingress.class: nginx
spec:
  ingressClassName: nginx
  rules:
  - host: www.wang.org
    http:
      paths:
      - backend:
          service:
            #name: pod-test-v1
            name: pod-test-v2          #修改此行,指向新版应用的SVC
            port:
              number: 80
          path: /
          pathType: Prefix
```

#应用

```
[root@master1 ~]#kubectl apply -f ingress-pod-test.yaml
```

#集群外客户端访问,观察全部都切换至新版本

```
[root@ubuntu2004 ~]#while true;do curl -H"host:www.wang.org"
http://10.0.0.99/;sleep 0.5;done
kubernetes pod-test v0.2!! ClientIP: 10.244.2.37, ServerName: pod-test-v2-
cfc768546-89z2g, ServerIP: 10.244.2.39!
kubernetes pod-test v0.2!! ClientIP: 10.244.2.37, ServerName: pod-test-v2-
cfc768546-89z2g, ServerIP: 10.244.2.39!
kubernetes pod-test v0.2!! ClientIP: 10.244.2.37, ServerName: pod-test-v2-
cfc768546-89z2g, ServerIP: 10.244.2.39!
kubernetes pod-test v0.2!! ClientIP: 10.244.2.37, ServerName: pod-test-v2-
cfc768546-89z2g, ServerIP: 10.244.2.39!
kubernetes pod-test v0.2!! ClientIP: 10.244.2.37, ServerName: pod-test-v2-
cfc768546-89z2g, ServerIP: 10.244.2.39!
kubernetes pod-test v0.2!! ClientIP: 10.244.2.37, ServerName: pod-test-v2-
cfc768546-89z2g, ServerIP: 10.244.2.39!
kubernetes pod-test v0.2!! ClientIP: 10.244.2.37, ServerName: pod-test-v2-
cfc768546-89z2g, ServerIP: 10.244.2.39!
kubernetes pod-test v0.2!! ClientIP: 10.244.2.37, ServerName: pod-test-v2-
cfc768546-89z2g, ServerIP: 10.244.2.39!
kubernetes pod-test v0.2!! ClientIP: 10.244.2.37, ServerName: pod-test-v2-
cfc768546-89z2g, ServerIP: 10.244.2.39!
```

#回滚旧版本

```
[root@master1 ~]#vim ingress-pod-test.yaml
```

```

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: pod-test
  annotations:
    kubernetes.io/ingress.class: nginx
spec:
  rules:
  - host: www.wang.org
    http:
      paths:
      - backend:
          service:
            #name: pod-test-v2
            name: pod-test-v1          #指向旧版应用的SVC
            port:
              number: 80
          path: /
          pathType: Prefix

#应用
[root@master1 ~]#kubectl apply -f ingress-pod-test.yaml

```

### 5.2.3 范例：基于权重的金丝雀发布

```

#旧版应用
[root@master1 ~]#kubectl apply -f ingress-pod-test.yaml

#新版的清单文件
[root@master1 ~]#cat canary-by-weight.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    #kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/canary: "true"
    nginx.ingress.kubernetes.io/canary-weight: "10"          #指定使用金丝雀发布新版占用的百分比10
  name: pod-test-canary-by-weight
spec:
  ingressClassName: nginx
  rules:
  - host: www.wang.org
    http:
      paths:
      - backend:
          service:
            name: pod-test-v2
            port:
              number: 80
          path: /
          pathType: Prefix

#应用
[root@master1 ~]#kubectl apply -f canary-by-weight.yaml

#集群外客户端访问，观察新旧版本的比例

```

```
[root@ubuntu2004 ~]#while true;do curl -H"host: www.wang.org"
http://10.0.0.99/;sleep 0.5;done
kubernetes pod-test v0.1!! ClientIP: 10.244.2.37, ServerName: pod-test-v1-
6ddc47c8cd-1mcmh, ServerIP: 10.244.2.38!
kubernetes pod-test v0.1!! ClientIP: 10.244.2.37, ServerName: pod-test-v1-
6ddc47c8cd-1mcmh, ServerIP: 10.244.2.38!
kubernetes pod-test v0.1!! ClientIP: 10.244.2.37, ServerName: pod-test-v1-
6ddc47c8cd-1mcmh, ServerIP: 10.244.2.38!
kubernetes pod-test v0.1!! ClientIP: 10.244.2.37, ServerName: pod-test-v1-
6ddc47c8cd-1mcmh, ServerIP: 10.244.2.38!
kubernetes pod-test v0.1!! ClientIP: 10.244.2.37, ServerName: pod-test-v1-
6ddc47c8cd-1mcmh, ServerIP: 10.244.2.38!
kubernetes pod-test v0.1!! ClientIP: 10.244.2.37, ServerName: pod-test-v1-
6ddc47c8cd-1mcmh, ServerIP: 10.244.2.38!
kubernetes pod-test v0.1!! ClientIP: 10.244.2.37, ServerName: pod-test-v1-
6ddc47c8cd-1mcmh, ServerIP: 10.244.2.38!
kubernetes pod-test v0.1!! ClientIP: 10.244.2.37, ServerName: pod-test-v1-
6ddc47c8cd-1mcmh, ServerIP: 10.244.2.38!
kubernetes pod-test v0.1!! ClientIP: 10.244.2.37, ServerName: pod-test-v1-
6ddc47c8cd-1mcmh, ServerIP: 10.244.2.38!
kubernetes pod-test v0.2!! ClientIP: 10.244.2.37, ServerName: pod-test-v2-
cfc768546-89z2g, ServerIP: 10.244.2.39!
kubernetes pod-test v0.1!! ClientIP: 10.244.2.37, ServerName: pod-test-v1-
6ddc47c8cd-1mcmh, ServerIP: 10.244.2.38!
kubernetes pod-test v0.1!! ClientIP: 10.244.2.37, ServerName: pod-test-v1-
6ddc47c8cd-1mcmh, ServerIP: 10.244.2.38!
kubernetes pod-test v0.1!! ClientIP: 10.244.2.37, ServerName: pod-test-v1-
6ddc47c8cd-1mcmh, ServerIP: 10.244.2.38!
kubernetes pod-test v0.1!! ClientIP: 10.244.2.37, ServerName: pod-test-v1-
6ddc47c8cd-1mcmh, ServerIP: 10.244.2.38!
kubernetes pod-test v0.2!! ClientIP: 10.244.2.37, ServerName: pod-test-v2-
cfc768546-89z2g, ServerIP: 10.244.2.39!
kubernetes pod-test v0.1!! ClientIP: 10.244.2.37, ServerName: pod-test-v1-
6ddc47c8cd-1mcmh, ServerIP: 10.244.2.38!
```

#调整weight权重为90

```
[root@master1 ~]#vim canary-by-weight.yaml
```

.....

```
    nginx.ingress.kubernetes.io/canary-weight: "90"
```

.....

#再应用

```
[root@master1 ~]#kubectl apply -f yaml/canary-by-weight.yaml
```

#观察比例变化

```
[root@ubuntu2004 ~]#while true;do curl -H"host:www.wang.org"
http://10.0.0.99/;sleep 0.5;done
```

```
kubernetes pod-test v0.2!! ClientIP: 10.244.2.37, ServerName: pod-test-v2-
cfc768546-89z2g, ServerIP: 10.244.2.39!
kubernetes pod-test v0.2!! ClientIP: 10.244.2.37, ServerName: pod-test-v2-
cfc768546-89z2g, ServerIP: 10.244.2.39!
kubernetes pod-test v0.2!! ClientIP: 10.244.2.37, ServerName: pod-test-v2-
cfc768546-89z2g, ServerIP: 10.244.2.39!
kubernetes pod-test v0.2!! ClientIP: 10.244.2.37, ServerName: pod-test-v2-
cfc768546-89z2g, ServerIP: 10.244.2.39!
```



```
kubernetes pod-test v0.2!! ClientIP: 10.244.2.37, ServerName: pod-test-v2-
cfc768546-89z2g, ServerIP: 10.244.2.39!
kubernetes pod-test v0.1!! ClientIP: 10.244.2.37, ServerName: pod-test-v1-
6ddc47c8cd-1mcmh, ServerIP: 10.244.2.38!
kubernetes pod-test v0.2!! ClientIP: 10.244.2.37, ServerName: pod-test-v2-
cfc768546-89z2g, ServerIP: 10.244.2.39!
kubernetes pod-test v0.1!! ClientIP: 10.244.2.37, ServerName: pod-test-v1-
6ddc47c8cd-1mcmh, ServerIP: 10.244.2.38!
kubernetes pod-test v0.2!! ClientIP: 10.244.2.37, ServerName: pod-test-v2-
cfc768546-89z2g, ServerIP: 10.244.2.39!
kubernetes pod-test v0.2!! ClientIP: 10.244.2.37, ServerName: pod-test-v2-
cfc768546-89z2g, ServerIP: 10.244.2.39!
kubernetes pod-test v0.2!! ClientIP: 10.244.2.37, ServerName: pod-test-v2-
cfc768546-89z2g, ServerIP: 10.244.2.39!
kubernetes pod-test v0.2!! ClientIP: 10.244.2.37, ServerName: pod-test-v2-
cfc768546-89z2g, ServerIP: 10.244.2.39!
kubernetes pod-test v0.2!! ClientIP: 10.244.2.37, ServerName: pod-test-v2-
cfc768546-89z2g, ServerIP: 10.244.2.39!
kubernetes pod-test v0.1!! ClientIP: 10.244.2.37, ServerName: pod-test-v1-
6ddc47c8cd-1mcmh, ServerIP: 10.244.2.38!
```

#如果需要回滚

```
[root@master1 ~]#kubectl edit ingress pod-test-canary-by-weight
nginx.ingress.kubernetes.io/canary-weight: "0" #修改新版比例为0,完全回滚
```

#环境清理

```
[root@master1 yam]#kubectl delete -f canary-by-weight.yaml
```

## 5.2.4 范例：基于 Cookie 实现金丝雀发布

#旧版应用

```
[root@master1 ~]#kubectl apply -f ingress-pod-test.yaml
```

#清单文件

```
[root@master1 ~]#cat canary-by-cookie.yaml
```

```
apiVersion: networking.k8s.io/v1
```

```
kind: Ingress
```

```
metadata:
```

```
  annotations:
```

```
    #kubernetes.io/ingress.class: nginx
```

```
    nginx.ingress.kubernetes.io/canary: "true"
```

```
    nginx.ingress.kubernetes.io/canary-by-cookie: "vip_user" #cookie中
```

**vip\_user=always**时才用金丝雀发布下面新版本

```
  name: pod-test-canary-by-cookie
```

```
spec:
```

```
  ingressClassName: nginx
```

```
  rules:
```

```
  - host: www.wang.org
```

```
    http:
```

```
      paths:
```

```
      - backend:
```

```
        service:
```

```
          name: pod-test-v2
```

```
          port:
```

```

        number: 80
    path: /
    pathType: Prefix

#应用
[root@master1 ~]#kubectl apply -f canary-by-cookie.yaml

#集群外客户端访问, 当cookie中有vip_user=always时才用金丝雀发布新版
[root@ubuntu2004 ~]#curl -H"host: www.wang.org" -b "vip_user=always"
http://10.0.0.99
kubernetes pod-test v0.2!! ClientIP: 10.244.2.37, ServerName: pod-test-v2-
cfc768546-89z2g, ServerIP: 10.244.2.39!

#集群外客户端访问, 当cookie中有vip_user!=always的其它值, 或没有vip_user时则不使用金丝雀发布
[root@ubuntu2004 ~]#curl -H"host: www.wang.org" -b "vip_user=never"
http://10.0.0.99
kubernetes pod-test v0.1!! ClientIP: 10.244.2.37, ServerName: pod-test-v1-
6ddc47c8cd-1mcmh, ServerIP: 10.244.2.38!
[root@ubuntu2004 ~]#curl -H"host: www.wang.org" -b "vip_user=wang"
http://10.0.0.99
kubernetes pod-test v0.1!! ClientIP: 10.244.2.37, ServerName: pod-test-v1-
6ddc47c8cd-1mcmh, ServerIP: 10.244.2.38!
[root@ubuntu2004 ~]#curl -H"host: www.wang.org" http://10.0.0.99
kubernetes pod-test v0.1!! ClientIP: 10.244.2.37, ServerName: pod-test-v1-
6ddc47c8cd-1mcmh, ServerIP: 10.244.2.38!

#清理环境
[root@master1 ~]#kubectl delete -f canary-by-cookie.yaml

```

## 5.2.5 范例：基于请求 Header 固定值的金丝雀发布

```

#旧版应用
[root@master1 ~]#kubectl apply -f ingress-pod-test.yaml

#清单文件
[root@master1 ~]#cat canary-by-header.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    #kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/canary: "true"
    nginx.ingress.kubernetes.io/canary-by-header: "X-Canary" #X-Canary自定义的首部
    字段值为always时才使用金丝雀发布下面新版本, 否则为旧版本
  name: pod-test-canary-by-header
spec:
  ingressClassName: nginx
  rules:
  - host: www.wang.org
    http:
      paths:
      - backend:
          service:
            name: pod-test-v2
            port:
              number: 80
        path: /

```

pathType: Prefix

#应用

```
[root@master1 ~]#kubectl apply -f canary-by-header.yaml
```

#查看

```
[root@master1 ~]#kubectl get ingress
```

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
pod-test	<none>	www.wang.org	10.0.0.99	80	12m
pod-test-canary-by-header	<none>	www.wang.org	10.0.0.99	80	53s

#集群外客户端访问，注意：always的小写的

#当X-Canary: always则访问v0.2新版

```
[root@ubuntu2004 ~]#curl -H"host:www.wang.org" -H"X-Canary: always"
http://10.0.0.99/
```

```
kubernetes pod-test v0.2!! ClientIP: 10.244.2.37, ServerName: pod-test-v2-
cfc768546-89z2g, ServerIP: 10.244.2.39!
```

#当X-Canary: never则访问v0.1旧版

```
[root@ubuntu2004 ~]#curl -H"host:www.wang.org" -H"X-Canary: never"
http://10.0.0.99/
```

```
kubernetes pod-test v0.1!! ClientIP: 10.244.2.37, ServerName: pod-test-v1-
6ddc47c8cd-1mcmh, ServerIP: 10.244.2.38!
```

#当X-Canary 是其它值则访问v0.1旧版

```
[root@ubuntu2004 ~]#curl -H"host:www.wang.org" http://10.0.0.99/
```

```
kubernetes pod-test v0.1!! ClientIP: 10.244.2.37, ServerName: pod-test-v1-
6ddc47c8cd-1mcmh, ServerIP: 10.244.2.38!
```

#当没有首部字段X-Canary则访问v0.1旧版

```
[root@ubuntu2004 ~]#curl -H"host:www.wang.org" -H"X-Canary: test"
http://10.0.0.99/
```

```
kubernetes pod-test v0.1!! ClientIP: 10.244.2.37, ServerName: pod-test-v1-
6ddc47c8cd-1mcmh, ServerIP: 10.244.2.38!
```

#清理环境

```
[root@master1 ~]#kubectl delete -f canary-by-header.yaml
```

## 5.2.6 范例：基于请求 Header 精确匹配指定值的金丝雀发布

#旧版应用

```
[root@master1 ~]#kubectl apply -f ingress-pod-test.yaml
```

#清单文件

```
[root@master1 ~]#cat canary-by-header-value.yaml
```

```
apiVersion: networking.k8s.io/v1
```

```
kind: Ingress
```

```
metadata:
```

```
  annotations:
```

```
    #kubernetes.io/ingress.class: nginx
```

```
    nginx.ingress.kubernetes.io/canary: "true"
```

```
    nginx.ingress.kubernetes.io/canary-by-header: "ISVIP" #字段名大小写不敏感
```

```
    nginx.ingress.kubernetes.io/canary-by-header-value: "true" #ISVIP首部字段的值为
```

true就使用金丝雀发布下面新版本,否则为旧版本,值大小写敏感

```
  name: pod-test-canary-by-header-value
```

```
spec:
```

```

ingressClassName: nginx
rules:
- host: www.wang.org
  http:
    paths:
    - backend:
        service:
          name: pod-test-v2
          port:
            number: 80
      path: /
      pathType: Prefix

```

#应用

```
[root@master1 ~]#kubectl apply -f canary-by-header-value.yaml
```

#集群外客户端访问，注意：**IsVIP**字段名大小写不敏感，**true**值大小写敏感

#当**IsVIP: true**，则访问v0.2新版

```
[root@ubuntu2004 ~]#curl -H"host:www.wang.org" -H"IsVIP: true" http://10.0.0.99/
kubernetes pod-test v0.2!! ClientIP: 10.244.2.37, ServerName: pod-test-v2-
cfc768546-89z2g, ServerIP: 10.244.2.39!
```

#当**IsVIP: false**，则访问v0.1旧版

```
[root@ubuntu2004 ~]#curl -k -H"host:www.wang.org" -H"IsVIP: false"
https://10.0.0.99/
kubernetes pod-test v0.1!! ClientIP: 10.244.2.37, ServerName: pod-test-v1-
6ddc47c8cd-1mcmh, ServerIP: 10.244.2.38!
```

#当**IsVIP: 其它值**，则访问v0.1旧版

```
[root@ubuntu2004 ~]#curl -k -H"host:www.wang.org" -H"IsVIP: test"
https://10.0.0.99/
kubernetes pod-test v0.1!! ClientIP: 10.244.2.37, ServerName: pod-test-v1-
6ddc47c8cd-1mcmh, ServerIP: 10.244.2.38!
```

#当没有首部字段**IsVIP**，则访问v0.1旧版

```
[root@ubuntu2004 ~]#curl -H"host:www.wang.org" http://10.0.0.99/
kubernetes pod-test v0.1!! ClientIP: 10.244.2.37, ServerName: pod-test-v1-
6ddc47c8cd-1mcmh, ServerIP: 10.244.2.38!
```

#清理环境

```
[root@master1 ~]#kubectl delete -f canary-by-header-value.yaml
```

## 5.2.7 范例：基于请求 Header 正则表达式模式匹配的指定值的金丝雀发布

#旧版应用

```
[root@master1 ~]#kubectl apply -f ingress-pod-test.yaml
```

#清单文件

```
[root@master1 ~]#cat canary-by-header-pattern.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    #kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/canary: "true"

```

```

    nginx.ingress.kubernetes.io/canary-by-header: "Username" #字段名大小不敏感
    nginx.ingress.kubernetes.io/canary-by-header-pattern: "^(vip|VIP)_.*" #首部字
段有Username且正则匹配时使用新版,否则为旧版本
    name: pod-test-canary-by-header-pattern
spec:
  ingressClassName: nginx
  rules:
  - host: www.wang.org
    http:
      paths:
      - backend:
          service:
            name: pod-test-v2
            port:
              number: 80
          path: /
          pathType: Prefix
#应用
[root@master1 ~]#kubectl apply -f canary-by-header-pattern.yaml

#集群外客户端访问
#符合正则表达式访问,则访问v0.2新版,字段名Username大小
[root@ubuntu2004 ~]#curl -H"host:www.wang.org" -H"Username: VIP_wang"
http://10.0.0.99/
kubernetes pod-test v0.2!! ClientIP: 10.244.2.37, ServerName: pod-test-v2-
cfc768546-89z2g, ServerIP: 10.244.2.39!
[root@ubuntu2004 ~]#curl -H"host:www.wang.org" -H"Username: VIP_test"
http://10.0.0.99/
kubernetes pod-test v0.2!! ClientIP: 10.244.2.37, ServerName: pod-test-v2-
cfc768546-89z2g, ServerIP: 10.244.2.39!
[root@ubuntu2004 ~]#curl -H"host:www.wang.org" -H"Username: vip_wang"
http://10.0.0.99/
kubernetes pod-test v0.2!! ClientIP: 10.244.2.37, ServerName: pod-test-v2-
cfc768546-89z2g, ServerIP: 10.244.2.39!

#不符合正则表达式访问,则访问v0.1旧版
[root@ubuntu2004 ~]#curl -H"host:www.wang.org" -H"Username: wang"
http://10.0.0.99/
kubernetes pod-test v0.1!! ClientIP: 10.244.2.37, ServerName: pod-test-v1-
6ddc47c8cd-1mcmh, ServerIP: 10.244.2.38!

#当没有首部字段Username,则访问v0.1旧版
[root@ubuntu2004 ~]#curl -H"host:www.wang.org" http://10.0.0.99/
kubernetes pod-test v0.1!! ClientIP: 10.244.2.37, ServerName: pod-test-v1-
6ddc47c8cd-1mcmh, ServerIP: 10.244.2.38!

#清理环境
[root@master1 ~]#kubectl delete -f canary-by-header-pattern.yaml

```

## 5.2.8 范例：基于cookie和Header 同时存在的金丝雀发布比较优先级

```

#旧版应用
[root@master1 ~]#kubectl apply -f ingress-pod-test.yaml

#同时应用两个ingress的清单文件

```

```
[root@master1 ~]#kubectl apply -f canary-by-header-pattern.yaml -f canary-by-header-pattern.yaml
```

```
[root@master1 ~]#kubectl get ingress
```

NAME	CLASS	HOSTS	ADDRESS
pod-test	nginx	www.wang.org	10.0.0.10,10.0.0.99
pod-test-canary-by-cookie	nginx	www.wang.org	10.0.0.10,10.0.0.99
pod-test-canary-by-header-pattern	nginx	www.wang.org	10.0.0.10,10.0.0.99

#基于cookie访问，优先级低于header，所以不生效，访问旧版本

```
[root@ubuntu2404 ~]#curl -H"cookie: vip_user=always" www.wang.org
kubernetes pod-test v0.1!! ClientIP: 10.244.1.126, ServerName: pod-test-v1-644d84d5df-gv5x8, ServerIP: 10.244.4.141!
```

#基于header访问，优先级高于cookie，所以优先生效

```
[root@ubuntu2404 ~]#curl -H"username: VIP_" www.wang.org
kubernetes pod-test v0.2!! ClientIP: 10.244.1.126, ServerName: pod-test-v2-555cf59c67-9l5vx, ServerIP: 10.244.3.181!
```

#删除header的ingress

```
[root@master1 ~]#kubectl delete ingress pod-test-canary-by-header-pattern
```

#再基于cookie访问，可以访问新版本

```
[root@ubuntu2404 ~]#curl -H"cookie: vip_user=always" www.wang.org
kubernetes pod-test v0.2!! ClientIP: 10.244.1.126, ServerName: pod-test-v2-555cf59c67-9l5vx, ServerIP: 10.244.3.181!
```