**Ingress**

**FEATURE STATE:** Kubernetes v1.1 [beta]

Ingress 是对集群中服务的外部访问进行管理的 API 对象，典型的访问方式是 HTTP。

Ingress 可以提供负载均衡、SSL 终结和基于名称的虚拟托管。

术语

为了表达更加清晰，本指南定义了以下术语：

* 节点（Node）: Kubernetes 集群中其中一台工作机器，是集群的一部分。
* 集群（Cluster）: 一组运行由 Kubernetes 管理的容器化应用程序的节点。 在此示例和在大多数常见的 Kubernetes 部署环境中，集群中的节点都不在公共网络中。
* 边缘路由器（Edge router）: 在集群中强制执行防火墙策略的路由器（router）。可以是由云提供商管理的网关，也可以是物理硬件。
* 集群网络（Cluster network）: 一组逻辑的或物理的连接，根据 Kubernetes [网络模型](https://kubernetes.io/zh/docs/concepts/cluster-administration/networking/) 在集群内实现通信。
* 服务（Service）：Kubernetes [服务](https://kubernetes.io/zh/docs/concepts/services-networking/service/)使用 [标签](https://kubernetes.io/zh/docs/concepts/overview/working-with-objects/labels/) 选择算符（selectors）标识的一组 Pod。 除非另有说明，否则假定服务只具有在集群网络中可路由的虚拟 IP。

Ingress 是什么？

[Ingress](https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.19/#ingress-v1beta1-networking-k8s-io) 公开了从集群外部到集群内[服务](https://kubernetes.io/zh/docs/concepts/services-networking/service/)的 HTTP 和 HTTPS 路由。 流量路由由 Ingress 资源上定义的规则控制。

internet

|

[ Ingress ]

--|-----|--

[ Services ]

可以将 Ingress 配置为服务提供外部可访问的 URL、负载均衡流量、终止 SSL/TLS，以及提供基于名称的虚拟主机等能力。 [Ingress 控制器](https://kubernetes.io/zh/docs/concepts/services-networking/ingress-controllers) 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](https://kubernetes.io/zh/docs/concepts/services-networking/service/#nodeport) 或 [Service.Type=LoadBalancer](https://kubernetes.io/zh/docs/concepts/services-networking/service/#loadbalancer) 类型的服务。

环境准备

你必须具有 [Ingress 控制器](https://kubernetes.io/zh/docs/concepts/services-networking/ingress-controllers) 才能满足 Ingress 的要求。 仅创建 Ingress 资源本身没有任何效果。

你可能需要部署 Ingress 控制器，例如 [ingress-nginx](https://kubernetes.github.io/ingress-nginx/deploy/)。 你可以从许多 [Ingress 控制器](https://kubernetes.io/zh/docs/concepts/services-networking/ingress-controllers) 中进行选择。

理想情况下，所有 Ingress 控制器都应符合参考规范。但实际上，不同的 Ingress 控制器操作略有不同。

**说明：** 确保您查看了 Ingress 控制器的文档，以了解选择它的注意事项。

Ingress 资源

一个最小的 Ingress 资源示例：

**apiVersion**: networking.k8s.io/v1beta1

**kind**: Ingress

**metadata**:

**name**: test-ingress

**annotations**:

**nginx.ingress.kubernetes.io/rewrite-target**: /

**spec**:

**rules**:

- **http**:

**paths**:

- **path**: /testpath

**pathType**: Prefix

**backend**:

**serviceName**: test

**servicePort**: 80

与所有其他 Kubernetes 资源一样，Ingress 需要使用 apiVersion、kind 和 metadata 字段。 Ingress 对象的命名必须是合法的 [DNS 子域名名称](https://kubernetes.io/zh/docs/concepts/overview/working-with-objects/names#dns-subdomain-names)。 有关使用配置文件的一般信息，请参见[部署应用](https://kubernetes.io/zh/docs/tasks/run-application/run-stateless-application-deployment/)、 [配置容器](https://kubernetes.io/zh/docs/tasks/configure-pod-container/configure-pod-configmap/)、 [管理资源](https://kubernetes.io/zh/docs/concepts/cluster-administration/manage-deployment/)。 Ingress 经常使用注解（annotations）来配置一些选项，具体取决于 Ingress 控制器，例如 [重写目标注解](https://github.com/kubernetes/ingress-nginx/blob/master/docs/examples/rewrite/README.md)。 不同的 [Ingress 控制器](https://kubernetes.io/zh/docs/concepts/services-networking/ingress-controllers) 支持不同的注解。查看文档以供您选择 Ingress 控制器，以了解支持哪些注解。

Ingress [规约](https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status) 提供了配置负载均衡器或者代理服务器所需的所有信息。 最重要的是，其中包含与所有传入请求匹配的规则列表。 Ingress 资源仅支持用于转发 HTTP 流量的规则。

Ingress 规则

每个 HTTP 规则都包含以下信息：

* 可选主机。在此示例中，未指定主机，因此该规则适用于通过指定 IP 地址的所有入站 HTTP 通信。 如果提供了主机（例如 foo.bar.com），则规则适用于该主机。
* 路径列表（例如，/testpath）,每个路径都有一个由 serviceName 和 servicePort 定义的关联后端。 在负载均衡器将流量定向到引用的服务之前，主机和路径都必须匹配传入请求的内容。
* 后端是 [Service 文档](https://kubernetes.io/zh/docs/concepts/services-networking/service/)中所述的服务和端口名称的组合。 与规则的主机和路径匹配的对 Ingress 的 HTTP（和 HTTPS ）请求将发送到列出的后端。

通常在 Ingress 控制器中会配置默认后端，以服务任何不符合规范中路径的请求。

默认后端

没有规则的 Ingress 将所有流量发送到同一个默认后端。 默认后端通常是 [Ingress 控制器](https://kubernetes.io/zh/docs/concepts/services-networking/ingress-controllers) 的配置选项，并且未在 Ingress 资源中指定。

如果主机或路径都没有与 Ingress 对象中的 HTTP 请求匹配，则流量将路由到默认后端。

路径类型

Ingress 中的每个路径都有对应的路径类型。当前支持的路径类型有三种：

* *ImplementationSpecific* （默认）：对于这种类型，匹配取决于 IngressClass。 具体实现可以将其作为单独的 pathType 处理或者与 Prefix 或 Exact 类型作相同处理。
* *Exact*：精确匹配 URL 路径，且对大小写敏感。
* *Prefix*：基于以 / 分隔的 URL 路径前缀匹配。匹配对大小写敏感，并且对路径中的元素逐个完成。 路径元素指的是由 / 分隔符分隔的路径中的标签列表。 如果每个 *p* 都是请求路径 *p* 的元素前缀，则请求与路径 *p* 匹配。

**说明：** 如果路径的最后一个元素是请求路径中最后一个元素的子字符串，则不会匹配 （例如：/foo/bar 匹配 /foo/bar/baz, 但不匹配 /foo/barbaz）。

多重匹配

在某些情况下，Ingress 中的多条路径会匹配同一个请求。 这种情况下最长的匹配路径优先。 如果仍然有两条同等的匹配路径，则精确路径类型优先于前缀路径类型。

Ingress 类

Ingress 可以由不同的控制器实现，通常使用不同的配置。 每个 Ingress 应当指定一个类，也就是一个对 IngressClass 资源的引用。 IngressClass 资源包含额外的配置，其中包括应当实现该类的控制器名称。

**apiVersion**: networking.k8s.io/v1beta1

**kind**: IngressClass

**metadata**:

**name**: external-lb

**spec**:

**controller**: example.com/ingress-controller

**parameters**:

**apiGroup**: k8s.example.com/v1alpha

**kind**: IngressParameters

**name**: external-lb

IngressClass 资源包含一个可选的参数字段，可用于为该类引用额外配置。

废弃的注解

在 Kubernetes 1.18 版本引入 IngressClass 资源和 ingressClassName 字段之前， Ingress 类是通过 Ingress 中的一个 kubernetes.io/ingress.class 注解来指定的。 这个注解从未被正式定义过，但是得到了 Ingress 控制器的广泛支持。

Ingress 中新的 ingressClassName 字段是该注解的替代品，但并非完全等价。 该注解通常用于引用实现该 Ingress 的控制器的名称， 而这个新的字段则是对一个包含额外 Ingress 配置的 IngressClass 资源的引用， 包括 Ingress 控制器的名称。

默认 Ingress 类

您可以将一个特定的 IngressClass 标记为集群默认选项。 将一个 IngressClass 资源的 ingressclass.kubernetes.io/is-default-class 注解设置为 true 将确保新的未指定 ingressClassName 字段的 Ingress 能够分配为这个默认的 IngressClass.

**注意：** 如果集群中有多个 IngressClass 被标记为默认，准入控制器将阻止创建新的未指定 ingressClassName 的 Ingress 对象。 解决这个问题只需确保集群中最多只能有一个 IngressClass 被标记为默认。

Ingress 类型

单服务 Ingress

现有的 Kubernetes 概念允许您暴露单个 Service (查看[替代方案](https://kubernetes.io/zh/docs/concepts/services-networking/ingress/#alternatives))。 你也可以通过指定无规则的 *默认后端* 来对 Ingress 进行此操作。

[service/networking/ingress.yaml](https://raw.githubusercontent.com/kubernetes/website/master/content/zh/examples/service/networking/ingress.yaml)

**apiVersion**: networking.k8s.io/v1beta1

**kind**: Ingress

**metadata**:

**name**: test-ingress

**spec**:

**backend**:

**serviceName**: testsvc

**servicePort**: 80

如果使用 kubectl apply -f 创建它，则应该能够查看刚刚添加的 Ingress 的状态：

kubectl get ingress test-ingress

NAME HOSTS ADDRESS PORTS AGE

test-ingress \* 203.0.113.123 80 59s

其中 203.0.113.123 是由 Ingress 控制器分配以满足该 Ingress 的 IP。

**说明：** 入口控制器和负载平衡器可能需要一两分钟才能分配 IP 地址。在此之前，您通常会看到地址字段的值被设定为 <pending>。

简单分列

一个分列配置根据请求的 HTTP URI 将流量从单个 IP 地址路由到多个服务。 Ingress 允许您将负载均衡器的数量降至最低。例如，这样的设置：

foo.bar.com -> 178.91.123.132 -> / foo service1:4200

/ bar service2:8080

将需要一个如下所示的 Ingress：

**apiVersion**: networking.k8s.io/v1beta1

**kind**: Ingress

**metadata**:

**name**: simple-fanout-example

**annotations**:

**nginx.ingress.kubernetes.io/rewrite-target**: /

**spec**:

**rules**:

- **host**: foo.bar.com

**http**:

**paths**:

- **path**: /foo

**backend**:

**serviceName**: service1

**servicePort**: 4200

- **path**: /bar

**backend**:

**serviceName**: service2

**servicePort**: 8080

当你使用 kubectl apply -f 创建 Ingress 时：

kubectl describe ingress simple-fanout-example

Name: simple-fanout-example

Namespace: default

Address: 178.91.123.132

Default backend: default-http-backend:80 (10.8.2.3:8080)

Rules:

Host Path Backends

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

foo.bar.com

/foo service1:4200 (10.8.0.90:4200)

/bar service2:8080 (10.8.0.91:8080)

Annotations:

nginx.ingress.kubernetes.io/rewrite-target: /

Events:

Type Reason Age From Message

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

Normal ADD 22s loadbalancer-controller default/test

Ingress 控制器将提供实现特定的负载均衡器来满足 Ingress，只要 Service (service1，service2) 存在。 当它这样做了，你会在地址字段看到负载均衡器的地址。

**说明：** 取决于你使用的 [Ingress 控制器](https://kubernetes.io/zh/docs/concepts/services-networking/ingress-controllers)， 你可能需要创建默认 HTTP 后端[服务](https://kubernetes.io/zh/docs/concepts/services-networking/service/)。

基于名称的虚拟托管

基于名称的虚拟主机支持将针对多个主机名的 HTTP 流量路由到同一 IP 地址上。

foo.bar.com --| |-> foo.bar.com service1:80

| 178.91.123.132 |

bar.foo.com --| |-> bar.foo.com service2:80

以下 Ingress 让后台负载均衡器基于[host 头部字段](https://tools.ietf.org/html/rfc7230#section-5.4) 来路由请求。

**apiVersion**: networking.k8s.io/v1beta1

**kind**: Ingress

**metadata**:

**name**: name-virtual-host-ingress

**spec**:

**rules**:

- **host**: foo.bar.com

**http**:

**paths**:

- **backend**:

**serviceName**: service1

**servicePort**: 80

- **host**: bar.foo.com

**http**:

**paths**:

- **backend**:

**serviceName**: service2

**servicePort**: 80

如果您创建的 Ingress 资源没有规则中定义的任何主机，则可以匹配指向 Ingress 控制器 IP 地址 的任何网络流量，而无需基于名称的虚拟主机。

例如，以下 Ingress 资源会将 first.bar.com 请求的流量路由到 service1， 将 second.foo.com 请求的流量路由到 service2， 而没有在请求中定义主机名的 IP 地址的流量路由（即，不提供请求标头）到 service3。

**apiVersion**: networking.k8s.io/v1beta1

**kind**: Ingress

**metadata**:

**name**: name-virtual-host-ingress

**spec**:

**rules**:

- **host**: first.bar.com

**http**:

**paths**:

- **backend**:

**serviceName**: service1

**servicePort**: 80

- **host**: second.foo.com

**http**:

**paths**:

- **backend**:

**serviceName**: service2

**servicePort**: 80

- **http**:

**paths**:

- **backend**:

**serviceName**: service3

**servicePort**: 80

TLS

你可以通过设定包含 TLS 私钥和证书的[Secret](https://kubernetes.io/zh/docs/concepts/configuration/secret/) 来保护 Ingress。 目前，Ingress 只支持单个 TLS 端口 443，并假定 TLS 终止。

如果 Ingress 中的 TLS 配置部分指定了不同的主机，那么它们将根据通过 SNI TLS 扩展指定的主机名 （如果 Ingress 控制器支持 SNI）在同一端口上进行复用。 TLS Secret 必须包含名为 tls.crt 和 tls.key 的键名。 这些数据包含用于 TLS 的证书和私钥。例如：

**apiVersion**: v1

**kind**: Secret

**metadata**:

**name**: testsecret-tls

**namespace**: default

**data**:

**tls.crt**: base64 encoded cert

**tls.key**: base64 encoded key

**type**: kubernetes.io/tls

在 Ingress 中引用此 Secret 将会告诉 Ingress 控制器使用 TLS 加密从客户端到负载均衡器的通道。 你需要确保创建的 TLS Secret 来自包含 sslexample.foo.com 的公用名称（CN）的证书。 这里的公共名称也被称为全限定域名（FQDN）。

**apiVersion**: networking.k8s.io/v1beta1

**kind**: Ingress

**metadata**:

**name**: tls-example-ingress

**spec**:

**tls**:

- **hosts**:

- sslexample.foo.com

**secretName**: testsecret-tls

**rules**:

- **host**: sslexample.foo.com

**http**:

**paths**:

- **path**: /

**backend**:

**serviceName**: service1

**servicePort**: 80

**说明：** 各种 Ingress 控制器所支持的 TLS 功能之间存在差异。请参阅有关 [nginx](https://kubernetes.github.io/ingress-nginx/user-guide/tls/)、 [GCE](https://git.k8s.io/ingress-gce/README.md#frontend-https) 或者任何其他平台特定的 Ingress 控制器的文档，以了解 TLS 如何在你的环境中工作。

负载均衡

Ingress 控制器启动引导时使用一些适用于所有 Ingress 的负载均衡策略设置， 例如负载均衡算法、后端权重方案和其他等。 更高级的负载均衡概念（例如持久会话、动态权重）尚未通过 Ingress 公开。 你可以通过用于服务的负载均衡器来获取这些功能。

值得注意的是，即使健康检查不是通过 Ingress 直接暴露的，在 Kubernetes 中存在并行概念，比如[就绪检查](https://kubernetes.io/zh/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/) 允许你实现相同的目的。 请检查特定控制器的说明文档，以了解它们是怎样处理健康检查的 ( [nginx](https://git.k8s.io/ingress-nginx/README.md)， [GCE](https://git.k8s.io/ingress-gce/README.md#health-checks))。

更新 Ingress

要更新现有的 Ingress 以添加新的 Host，可以通过编辑资源来对其进行更新：

kubectl describe ingress test

Name: test

Namespace: default

Address: 178.91.123.132

Default backend: default-http-backend:80 (10.8.2.3:8080)

Rules:

Host Path Backends

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

foo.bar.com

/foo service1:80 (10.8.0.90:80)

Annotations:

nginx.ingress.kubernetes.io/rewrite-target: /

Events:

Type Reason Age From Message

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

Normal ADD 35s loadbalancer-controller default/test

kubectl edit ingress test

这一命令将打开编辑器，允许你以 YAML 格式编辑现有配置。 修改它来增加新的主机：

**spec**:

**rules**:

- **host**: foo.bar.com

**http**:

**paths**:

- **backend**:

**serviceName**: service1

**servicePort**: 80

**path**: /foo

- **host**: bar.baz.com

**http**:

**paths**:

- **backend**:

**serviceName**: service2

**servicePort**: 80

**path**: /foo

..

保存更改后，kubectl 将更新 API 服务器中的资源，该资源将告诉 Ingress 控制器重新配置负载均衡器。

验证：

kubectl describe ingress test

Name: test

Namespace: default

Address: 178.91.123.132

Default backend: default-http-backend:80 (10.8.2.3:8080)

Rules:

Host Path Backends

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

foo.bar.com

/foo service1:80 (10.8.0.90:80)

bar.baz.com

/foo service2:80 (10.8.0.91:80)

Annotations:

nginx.ingress.kubernetes.io/rewrite-target: /

Events:

Type Reason Age From Message

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

Normal ADD 45s loadbalancer-controller default/test

你也可以通过 kubectl replace -f 命令调用修改后的 Ingress yaml 文件来获得同样的结果。

跨可用区失败

不同的云厂商使用不同的技术来实现跨故障域的流量分布。详情请查阅相关 Ingress 控制器的文档。 请查看相关[Ingress 控制器](https://kubernetes.io/zh/docs/concepts/services-networking/ingress-controllers) 的文档以了解详细信息。 你还可以参考[联邦文档](https://github.com/kubernetes-sigs/federation-v2)，以获取有关在联合集群中部署 Ingress 的详细信息。

未来工作

跟踪 [SIG Network](https://github.com/kubernetes/community/tree/master/sig-network) 的活动以获得有关 Ingress 和相关资源演变的更多细节。 你还可以跟踪 [Ingress 仓库](https://github.com/kubernetes/ingress/tree/master) 以获取有关各种 Ingress 控制器的更多细节。

替代方案

不直接使用 Ingress 资源，也有多种方法暴露 Service：

* 使用 [Service.Type=LoadBalancer](https://kubernetes.io/docs/concepts/services-networking/service/#loadbalancer)
* 使用 [Service.Type=NodePort](https://kubernetes.io/docs/concepts/services-networking/service/#nodeport)

接下来

* 进一步了解 [Ingress API](https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.19/#ingress-v1beta1-networking-k8s-io)
* 进一步了解 [Ingress 控制器](https://kubernetes.io/zh/docs/concepts/services-networking/ingress-controllers/)
* [使用 NGINX 控制器在 Minikube 上安装 Ingress](https://kubernetes.io/zh/docs/tasks/access-application-cluster/ingress-minikube)