10 | K8s 极简实战（五）：如何将集群的业务服务暴露外网访问？

2022-12-30 王炜 来自北京

《云原生架构与GitOps实战》

课程介绍 >

讲述：王炜

时长 09:35　大小 8.76M

你好，我是王炜。

上一节课，我们学习了 Env、ConfigMap 和 Secret 三种管理应用配置的方法，了解了它们的适用场景。

当我们将应用顺利迁移到 Kubernetes 之后，接下来我们面临的第一个问题是，用户如何访问集群内部的业务服务呢？因为无论是 Pod 还是 Service，它的 IP 地址都是集群内的虚拟 IP 地址，也叫做 VIP，它实际上是一个内网 IP 地址，只能提供集群内的访问能力，并不能在公网环境下进行访问。

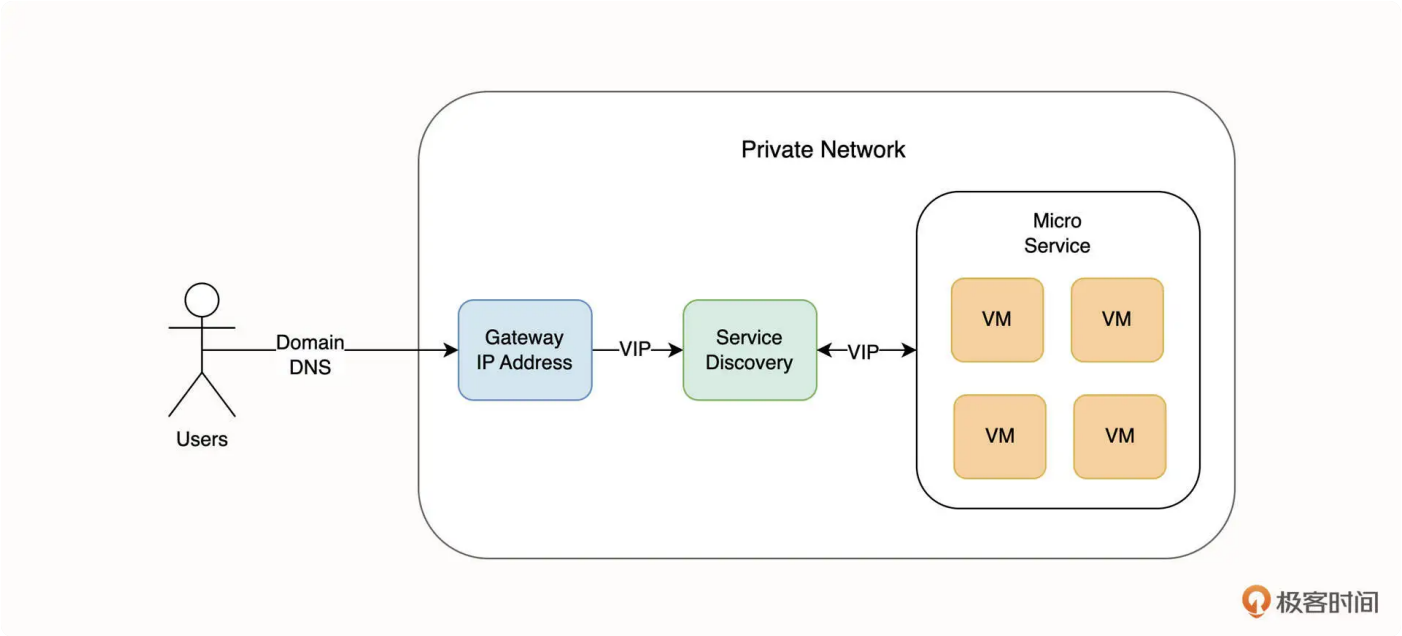
这节课，我们先来温习一下传统应用暴露公网的方式，然后结合示例应用，进一步学习如何对外暴露 Kubernetes 集群内的应用服务。

在开始之前，你需要确保已经按照第 5 讲“示例应用介绍”的引导在本地 Kind 集群部署了示例应用。

传统应用的服务暴露

我们先来回顾一下传统的微服务应用是如何对外暴露的。

一般来说，一个典型的微服务应用在系统的最外层会使用网关或者负载均衡器作为系统的入口，然后，根据路由规则和服务发现机制将流量转发到实际的后端微服务中（一般是业务进程所在的虚拟机上）。整体架构如下图所示。



在这个典型的微服务架构中，**网关是系统唯一的入口。**它将通过一个外网 IP 暴露业务系统，除了网关以外，整个业务系统的所有服务都在私有网络下，彼此通过 VIP 进行通信，外部无法访问除了网关以外的任何服务。通常，由于用户访问业务系统一般是使用域名，所以在网关前面还会有 DNS 解析步骤。

显然，在这种架构体系下，对外暴露业务只需要赋予网关服务一个公网 IP 地址就可以达到目的了。

Kubernetes 服务暴露

那么，在 Kubernetes 里面有没有类似的机制呢？结合 Kubernetes Service 对象，如果我们能赋予 Service 一个公网 IP，是不是就可以暴露 Service 选择器所关联的 Pod 了呢？

这个思路完全正确，但我们怎么才能让 Service 获得一个公网 IP 呢？

在回答这个问题之前，我想先请你回想一下第 8 讲的内容。在讲解如何解决服务发现问题时，我有提到过 Service 的两种类型，我有提到过 Service 的两种类型，**分别是 NodePort 和 Loadbalancer**。这两种类型都可以为 Service 赋予公网 IP。

接下来，我们深入了解一下这两种 Service 类型。

NodePort

当 Service 被配置为 NodePort 类型之后，Kubernetes 会在每一个节点上监听指定的端口（一般是 30000-32767），当通过节点的公网 IP + 端口号的形式访问时，请求会被转发到对应的 Service 当中。你可以理解为，NodePort 类型可以直接让 Service 在节点层面对外暴露。

在之前部署示例应用时，我们在本地为 Kind 集群安装了 Ingress-Nginx 组件，在 Kind 环境下，这个组件其实就是通过 NodePort 的方式暴露的。你可以通过 kubectl get service 来获取 Ingress-Nginx 的 Service Manifest。

复制代码

```
$ kubectl get service ingress-nginx-controller -n ingress-nginx -o yaml
apiVersion: v1
kind: Service
metadata:
  ......
  name: ingress-nginx-controller
  namespace: ingress-nginx
spec:
  ......
  ports:
  - appProtocol: http
    name: http
    nodePort: 31844
    port: 80
    protocol: TCP
    targetPort: http
  - appProtocol: https
    name: https
    nodePort: 32606
    port: 443
    protocol: TCP
    targetPort: https
  selector:
    app.kubernetes.io/component: controller
```

```
    app.kubernetes.io/instance: ingress-nginx
    app.kubernetes.io/name: ingress-nginx
  type: NodePort
```


在这段 Manifest 内容中，我们重点关注 Selector 字段，还有 Ports 字段下的 **NodePort、Port 和 TargetPort**。

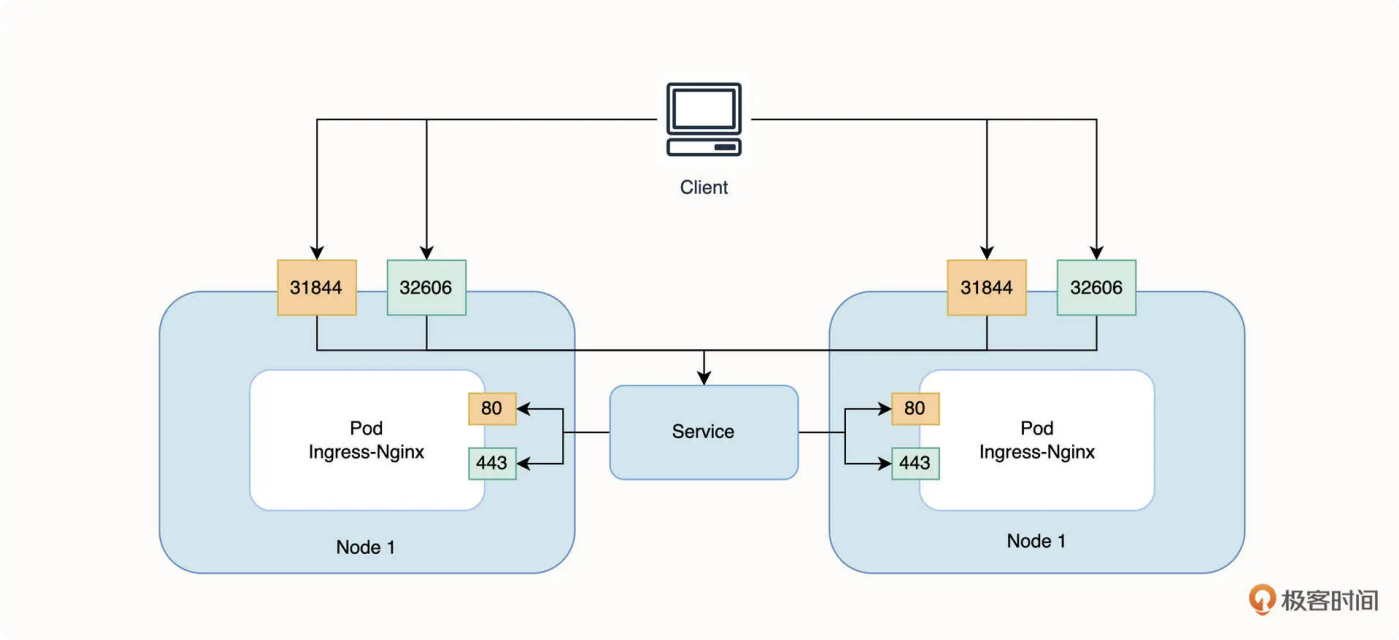
Selector 是选择器，它将匹配 Pod 模板里的 Labels，作用是抽象一组 Pod 服务并将流量在这些 Pod 中做负载均衡。

Ports 字段下定义了两个数组。

第一个数组中的 Port 字段代表 Service 的访问端口，你可以理解为，是 Service 在集群内部的暴露端口。TargetPort 表示目标端口，作用是告诉 Service 将请求转发到 Pod 的哪个端口，你可以理解为，是业务进程在容器里的监听端口。显然 Nginx 在容器内的监听端口是 80。最后也是最重要的 NodePort 字段，它表示需要将服务暴露在 Kubernetes 节点的什么端口，这里具体的含义是将服务暴露在 Kubernetes 节点的 31844 端口上。

同理，Ports 的第二个数组也是代表类似的含义。

最后，这段 Service Manifest 实现的效果是，访问 Kubernetes **任何一个节点**的公网 IP+31844 端口或 32606 端口，请求流量都会被转发到 Ingress-Nginx Pod 的 80 端口或 443 端口上。



NodePort 的暴露方式虽然可以直接复用 Kubernetes 节点的公网 IP，但我并**不推荐你在生产环境使用**它。主要的原因有两个。

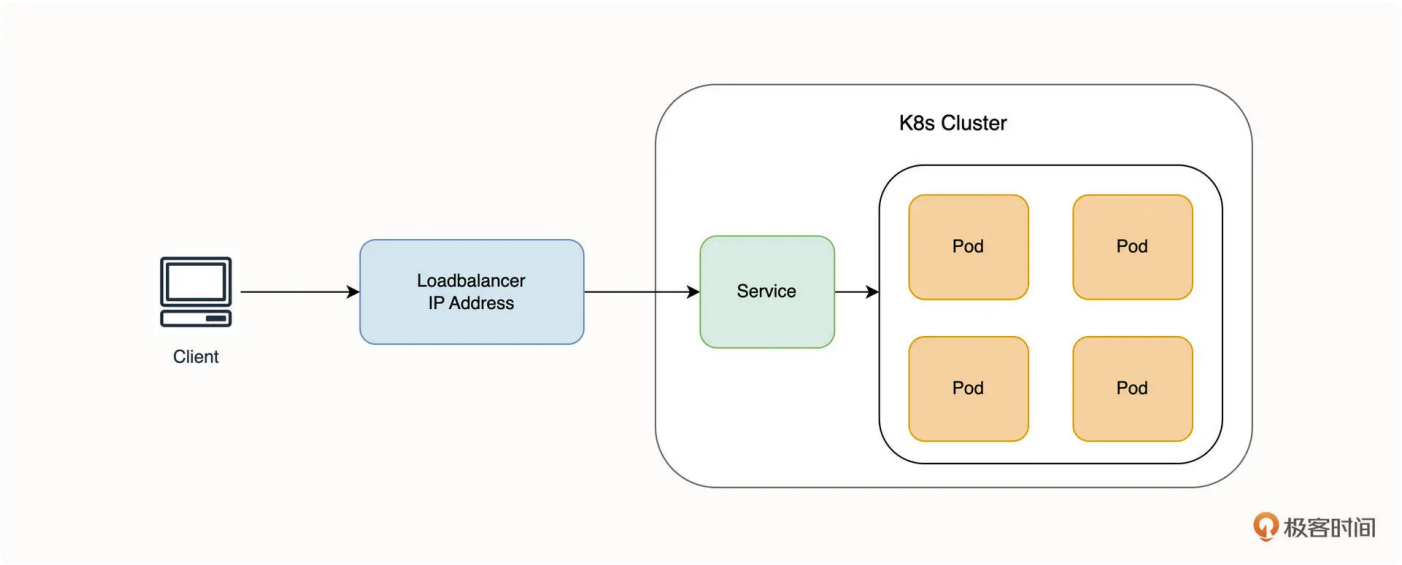
首先，直接对外暴露服务不利于统一管理外部请求流量。

其次，一个端口只能绑定一个服务，并且默认的端口范围是有限的，所以在较大规模场景时使用容易产生端口冲突。如果你希望临时访问集群内的业务服务，建议你使用端口转发进行访问，它适用于大多数的临时场景。

Loadbalancer

除了 NodePort 类型， Loadbalancer 类型也可以对外暴露 Service 服务，也就是我们常说的负载均衡器类型。

Loadbalancer 类型一般依赖于云厂商实现。当 Service 被声明为负载均衡器类型时，云厂商会创建一个负载均衡器实例并和集群的 Service 关联，借助负载均衡器的外网 IP 地址，实现 Service 的对外暴露。此时，相当于每一个 Loadbalancer 类型的 Service 都具有一个外网 IP 地址，所有流量先通过负载均衡器，再转发到对应的 Service 当中，如下图所示。



Loadbalancer 类型相比较 NodePort 有一定的优势。比如，理论上来说它暴露服务的数量不会受到端口数量的限制。从架构设计上来说，暴露服务和 Kubernetes 节点实现了解耦，是一个非常不错的选型。

需要注意的是，每声明一个 Loadbalancer 类型的 Service，都会创建一个新的负载均衡器实例，负载均衡器由于具有固定 IP 地址，所以费用也相对较高，并且还需要为流量额外付费。

所以，在实际的项目中，我们一般不直接用 Loadbalancer 类型对外暴露服务，而是通过**网关**来实现服务暴露，这和我们之前提到的传统应用的服务暴露方式非常类似。这时候，就不得不提到 Ingress 了。

Ingress

Ingress 是 Kubernetes 的一个内置对象，通常我们把 Ingress 看作 **Service 之上的 Service。**Ingress 对象只用来声明路由策略，并不处理具体的流量转发。要使得 Ingress 生效，我们还需要额外安装 Ingress-Controller，例如 Ingress-Nginx。

在生产环境中，Ingress-Nginx 一般就是以 Loadbalancer 类型来对外暴露的，Ingress-Nginx 实际上充当的是网关的角色，这样做的好处是，**我们只需要一个负载均衡器实例，通过路由策略，就可以对外暴露所有的业务服务。**

示例应用的 Ingress

接下来，我们通过示例应用来进一步理解 Ingress 对象。在部署示例应用时，我们已经为本地集群部署了 Ingress-Nginx，并且将 Ingress 对象部署到了集群中，你可以通过 kuebctl get ingress 来获取 Manifest。

复制代码

```
$ kubectl get ingress frontend-ingress -n example -o yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  ......
  name: frontend-ingress
  namespace: example
spec:
  ingressClassName: nginx
  rules:
  - http:
      paths:
      - backend:
          service:
            name: frontend-service
            port:
              number: 3000
        path: /?(.*)
        pathType: Prefix
      - backend:
          service:
            name: backend-service
```

```
            port:
              number: 5000
        path: /api/?(.*)
        pathType: Prefix
```


在这段 Ingress 配置中，我们重点关注 Paths 字段下的 Path 和 Backend 字段。

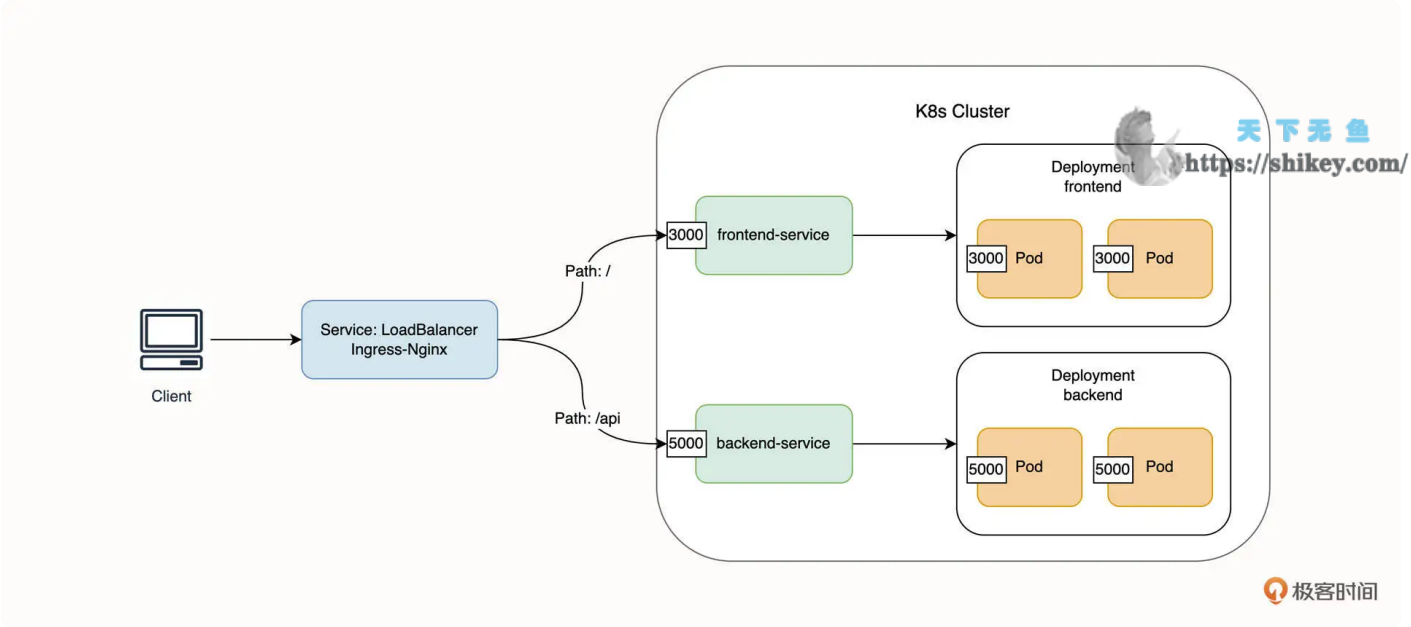
Paths 字段下有两个数组。第一个数组代表的路由策略是，当 URL 包含 / 前缀匹配时，那么将请求转发到 Backend 字段的配置的服务中，也就是转发到 frontend-service 的 3000 端口。同理，第二个数组的含义是，当 URL 包含 /api 前缀匹配时，那么将请求转发到 backend-service 的 5000 端口。

细心的你应该会发现，Ingress 指定的 Service 端口号其实就是 Service 对象的 Port 字段，这里我们可以结合 frontend-service 的内容进行对比。

复制代码

```
apiVersion: v1
kind: Service
metadata:
  name: frontend-service
spec:
  type: ClusterIP
  selector:
    app: frontend
  ports:
  - port: 3000
    targetPort: 3000
```

也就是说，以 Paths 第一个数组的路由策略为例，当 Service 接收到 Ingress 转发过来的流量之后，Service 会继续将流量转发到符合选择器 Labals app=frontend Pod 的 3000 端口上，这样就完成了一个完整的请求链路。为了让你更好地理解，我画了张流量的链路图，你可以结合图例来进行理解。



你需要额外注意一个细节，在本地的 Kind 测试集群和示例应用中，Ingress-Nginx 是通过 NodePort 的方式对外暴露的，这是因为我们在本地 Kind 集群中安装的是特殊版本的 Ingress-Nginx，而生产版本的 Ingress-Nginx 一般是通过 LoadBalancer 对外暴露的。

生产环境下部署 Ingress-Nginx

通常，在生产环境下，我们会使用云厂商直接提供的 Kubernetes 集群和部署生产版本的 Ingress-Nginx。要部署生产版本的 Ingress-Nginx 可以使用下面的命令。

```
$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/c
```

部署完成后，你可以通过 kubectl get svc 来获取 Ingress-Nginx 的外网 IP 地址。注意，需要指定 ingress-nginx 命名空间。

```
$ kubectl get svc -n ingress-nginx
NAME                                 TYPE           CLUSTER-IP     EXTERNAL-IP
ingress-nginx-controller             LoadBalancer   10.96.146.9    18.176.38.12
```

其中，EXTERNAL-IP 就是 Ingress-Nginx 的外网 IP 地址，这个 IP 也就是业务系统的唯一入口，我们可以将它配置到 DNS 域名解析的记录中，这样用户就可以通过域名的方式来访问业务应用了。

在多数实际项目中，Ingress 都是服务暴露的最佳实践。

总结

在这节课中，我们回顾了传统微服务应用的暴露方式。也就是，在最外层部署一个网关，把它作为业务系统的唯一入口。这种服务暴露方式具有很多优点，例如可以很方便地统一管理进入业务系统的流量，网关层面也可以做一些统一的认证和授权等。

在 Kubernetes 环境下，对外暴露服务则需要通过 Service 来实现。由于 Service 的 IP 默认是一个集群内的 IP，无法从外部访问，所以我们需要为 Service 赋予外网 IP 地址。其中，NodePort 类型可以通过 Kubernetes 节点外网 IP + 端口号的方式提供外网访问能力，不过并不推荐在生产环境使用这种方式。

LoadBalancer 类型则需要依赖云厂商的负载均衡器，一般由云厂商实现。在对 Service 配置为 LoadBalancer 类型后，云厂商将会异步创建负载均衡器实例，这种方式将暴露服务和 Kubernetes 节点进行了解耦，是一种常用的服务暴露方式。

不过在生产环境下，我们也不推荐以 LoadBalancer 的方式暴露所有需要在外网访问的业务服务，因为这不利于统一管理访问流量，并且还要为多个负载均衡器实例支付高昂的费用。为了解决这个问题，我们引入了 Ingress 来暴露服务。

值得注意的是，要使用 Ingress 除了声明 Ingress 对象以外，还需要为集群安装 Ingress-Controller，例如最常见的 Ingress-Nginx。在生产环境下，Ingress-Nginx 正是通过 LoadBalancer 类型的 Service 来自身暴露在公网环境的。

通过 Ingress 暴露服务的方式是我们在生产环境下最常用的方法，同时也是服务暴露的最佳实践。

思考题

最后，给你留两道思考题吧。

1. 你能简单分享目前项目使用的服务暴露方式吗？可以使用文字描述也可以尝试画一个架构图。

2. 请你将目前工作中业务服务的暴露方式和 Kubernetes Ingress 服务暴露方式做一个简单的比较。你认为它们的优劣势分别是什么？

欢迎你给我留言交流讨论，你也可以把这节课分享给更多的朋友一起阅读。我们下节课见。

上一篇 09丨K8s 极简实战（四）：如何迁移应用配置？

下一篇 11丨K8s 极简实战（六）：如何保障业务资源需求和自动弹性扩容？

精选留言 (11)

李多
2023-01-08 来自广东

这里推荐一个开源项目：MetalLB 。之前在做实验的时候，拉取的微服务yaml很多都用了LB。但是这些通常需要租用云厂商的k8s集群才能有。就找到了 MetalLB 这个项目，支持在本地集群、裸金属集群上面部署LB。在开发测试或者实验环境也能使用 LoadBalancer 类型的服务暴露了。我在用下来感觉配置也是比较简单，好上手。

https://github.com/metallb/metallb
https://metallb.universe.tf/

作者回复: 非常不错的项目～

共 2 条评论 >　2

GAC·DU
2022-12-30 来自广东

老师，域名只能绑在一个IP上，只使用ingress做为流量入口会造成单点故障。如果在ingress

前面再加一个LB，域名绑在LB上，利用LB将流量分发到ingress上，这样可以吗？

作者回复: 可以这么做。

不过在生产环境下，一般 ingress-nginx HA 我们会这么做：

1. 在几台高配节点上部署独占的实例，并配置相同的资源配额和限制，这样可以避免业务系统枪战资源以及 ingress-nginx 被驱逐。
2. 为 ingress-nginx 配置 HPA 策略

这样就可以得到高可用网关。

3

农民园丁
2023-01-14 来自内蒙古

在腾讯云搭建了1个master节点，2个node节点的实验环境。
➜ ingress git:(master) kubectl get svc -n ingress-nginx

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
ingress-nginx-controller LoadBalancer 10.97.238.89 <pending> 80:32446/TCP,443:31186/TCP 18m
ingress-nginx-controller-admission ClusterIP 10.105.63.105 <none> 443/TCP

外网IP始终是pending？
请问老师，ingress-nginx-controller 各个云服务商都是部署这个链接的yaml文件吗：
https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.3.1/deploy/static/provider/cloud/deploy.yaml

作者回复: 部署的 ingress 版本看起来没问题，可以尝试 kubectl describe service ingress-nginx-controller -n ingress-nginx 查看一下事件。另外可以去腾讯云控制台打开负载均衡器，检查实例初始化的情况。

共 4 条评论>

1

农民园丁
2023-01-12 来自内蒙古

如果是Kind创建的cluster，就是从宿主机的80端口-暴-露-的？

作者回复: 是的，前提是需要指定 config.yaml 暴露 80 端口。

1

小路飞😁

2023-01-30 来自新疆

老师你好，我这边的容器平台 gateway 采用traefik proxy组件
通过定义hosts 与 route 进项设置，实际上就是将ingress进行拆分成两个kind
前期工作中，服务需要对外访问，流程大致如下：内网 网关单点部署，绑定master 其中一个ip，网关端口为80与443，进行权限申请
以及白名单或者网络策略放通，master ip (80/443) --- > 公网ip (80/443) --- > 域名(80/443)
后续环境出现故障，内网绑定master 单点故障问题，后续启动了，原先集群的vip ip，vip ip 绑定了 三个 master ip，例如 172.xxx.xxx.88绑定172.xxx.xxx.83~85
首先升级网关多副本，master 单独标记一个网关标签，每个gateway都在master节点运行一个，访问流程：vip ip (80/443) --- > 公网ip (80/443) --- > 域名(80/443)
但是近期在适配的过程中，出现内网ip 可以正常访问，以及宿主机添加hosts 填写vip ip，绑定域名，进行访问业务流程均无问题，但是公网访问404 page not found
traefik 网关，在日常配置公网暴露服务时，直接通过定义主机以及路由，均是通过域名格式去定义，本地环境通过hosts手动绑定，域名访问策略生效。暂时有点怀疑
我的vip ip 后续升级可能存在问题，近期准备切换至以前单点模式，单个master ip --- > 公网 --- > 域名。

小路飞😁

2023-01-30 来自新疆

我这边的云平台，采用traefik

JDLYyear

2022-12-31 来自广东

老师，文中说：Ingress 对象只用来声明路由策略，并不处理具体的流量转发。
那流量转发实际上还是service来做吗？

作者回复: 这里我们安装的是 Ingress-Nginx，流量实际上先是由Nginx 转发到 Service 的，然后 Service 转发到业务 Pod 里的。

PeiHongbing

2022-12-30 来自广东

```
# kubectl -n ingress-nginx get svc
NAME                       TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)
AGE
ingress-nginx-controller   NodePort   10.96.133.249   <none>        80:32542/TCP,443:
32255/TCP   4d2h
```

为啥在节点查看端口32542和32255未被监听。

作者回复: Kind 比较特殊，他的节点是由 Docker 容器模拟的，你可以通过 docker ps 查看，然后进入容器里进一步检查看看。

PeiHongbing
2022-12-30 来自辽宁


```
# kubectl -n ingress-nginx get svc
NAME                       TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)
AGE
ingress-nginx-controller   NodePort   10.96.133.249   <none>        80:32542/TCP,443:
32255/TCP   4d2h
```


0ck0
2022-12-30 来自广东

生产环境下部署 Ingress-Nginx, 公网 ip 是如何分配给 Ingress-Nginx ？

作者回复: 在生产环境下部署的 ingress-nginx 是 LoadBalancer 类型的 Service，云厂商会自动创建负载均衡器并进行关联。

橙汁
2022-12-30 来自广东

文章中演示的在生产环境部署ingress，感觉大家不理解别纠结 看看云厂商的比如阿里 有组件部署后会直接与slb链接，自动生成IP地址，直接配置ingress规则即可，我猜老师演示的也是在云服务器上不然不会出来公网地址。

作者回复: 是的，生产环境下云厂商会实现 lb 类型的 ingress，所以就会有外网 IP，本地 kind 集群是没有的。

共 2 条评论>