60Kubernetes 系列（五十六）比较 Kubernetes Gateway 和 Ingress APIs

Kubernetes 系列（五十六）比较 Kubernetes Gateway 和 Ingress APIs

欢迎关注我的公众号" **云原生拓展** "，原创技术文章第一时间推送。

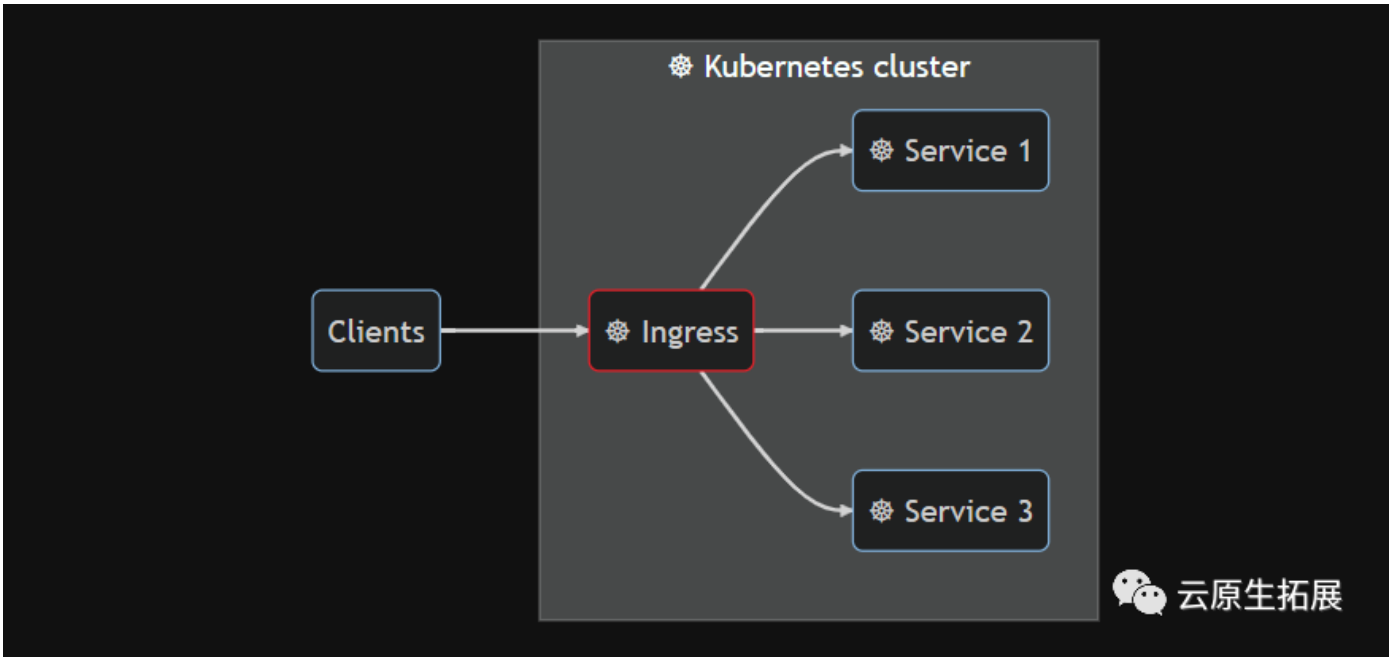
几个月前，新的 Kubernetes Gateway API 毕业为 beta 版本 。

当已经有稳定的 Kubernetes Ingress API 和 数十种实现 处理外部流量时，为什么需要另一个处理外部流量的 API 呢？新的 Gateway API 解决了 Ingress API 的哪些问题？这是否意味着 Ingress API 的终结？

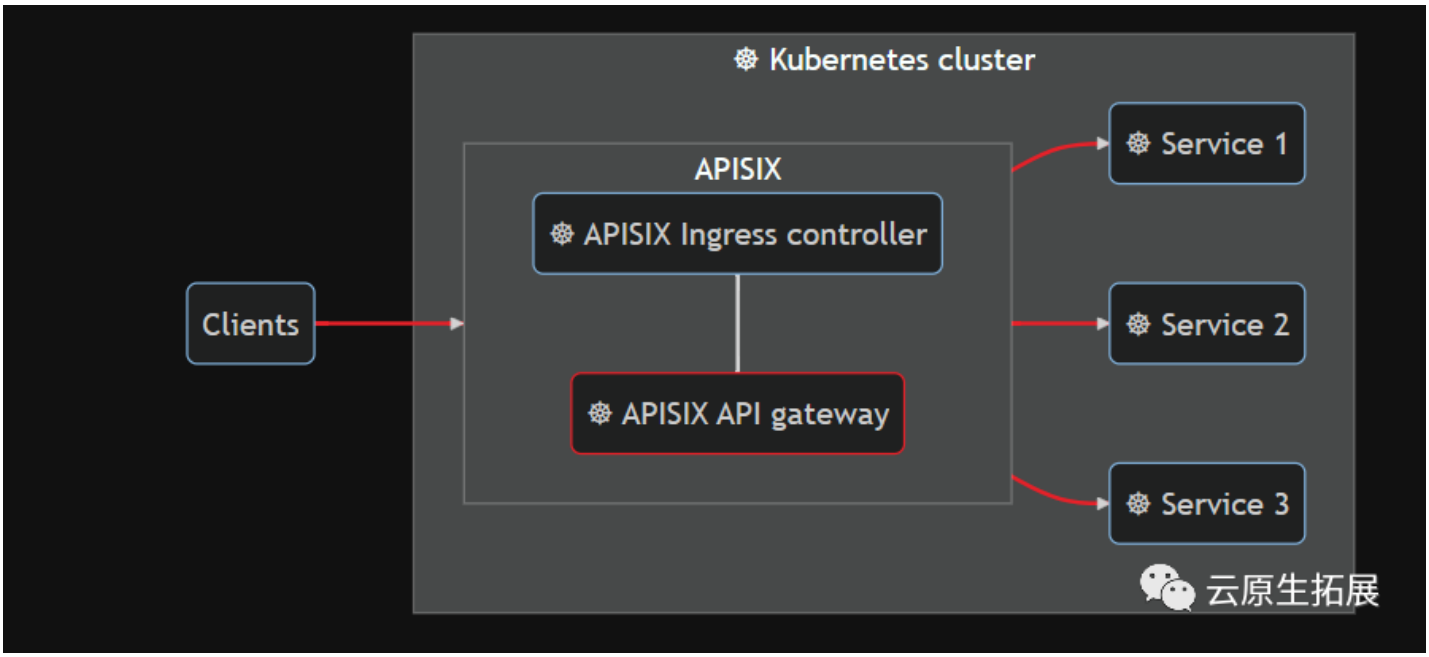
本文将通过对这些 API 进行实际操作并了解它们的发展来回答这些问题。

标准化服务的外部访问：Ingress API

Kubernetes Ingress API 的创建是为了标准化 Kubernetes 中将服务暴露给外部流量的方式。Ingress API 通过引入路由和 SSL 终止等功能，克服了默认服务类型 `NodePort` 和 `LoadBalancer` 的限制。



目前有超过 20种Ingress控制器的实现 。 在本文中，我将使用 Apache APISIX 及其 Ingress 控制器 作为示例。



您可以创建 Ingress 资源 来配置 Apache APISIX 或任何其他 Ingress 实现。

以下示例显示了如何使用 APISIX Ingress 在两个应用程序版本之间路由流量：

```yaml
kubernetes-ingress-manifest.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: api-routes
spec:
  ingressClassName: apisix
  rules:
    - host: local.navendu.me
      http:
        paths:
          - backend:
              service:
                name: bare-minimum-api-v1
                port:
                  number: 8080
            path: /v1
            pathType: Prefix
          - backend:
              service:
                name: bare-minimum-api-v2
                port:
                  number: 8081
            path: /v2
            pathType: Prefix
```

Ingress API 不绑定到任何特定的控制器实现，因此您可以将 APISIX 替换为任何其他 Ingress 控制器，它们将具有类似的工作方式。

这对于简单路由是可以的。但是该 API 有限，如果您想使用 Ingress 控制器提供的所有功能，您将被限制于使用 annotations 。

例如，Kubernetes Ingress API 不提供配置重写的模式。重写在您的上游/后端 URL 与 Ingress 规则中配置的路径不同时非常有用。

APISIX 支持此功能，您必须使用自定义注释才能利用它：

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: api-routes
  annotations:
    k8s.apisix.apache.org/rewrite-target-regex: "/app/(.*)"
    k8s.apisix.apache.org/rewrite-target-regex-template: "/$1"
spec:
  ingressClassName: apisix
  rules:
    - host: local.navendu.me
      http:
        paths:
          - backend:
              service:
                name: bare-minimum-api
                port:
                  number: 8080
            path: /app
            pathType: Prefix
```

这将创建一个 Ingress 资源，该资源配置了 APISIX，以便将具有 `/app` 前缀的任何请求路由到删除前缀的后端。例如，对 `/app/version` 的请求将被转发到 `/version` 。

注释特定于您选择的 Ingress 控制器。这些“专有”扩展限制了最初使用 Ingress API 意图的可移植性范围。

自定义 CRD > Ingress API

被注释限制的 Ingress 控制器也牺牲了 Ingress 控制器的可用性。

因此，控制器通过创建自己的 自定义资源 来解决 Ingress API 的限制。以下示例显示了配置 Ingress 以使用 APISIX 的自定义资源在两个应用程序版本之间路由流量：

```
apisix-ingress-manifest.yaml
apiVersion: apisix.apache.org/v2
kind: ApisixRoute
metadata:
  name: api-routes
spec:
  http:
    - name: route-1
      match:
        hosts:
          - local.navendu.me
        paths:
          - /v1
      backends:
        - serviceName: bare-minimum-api-v1
          servicePort: 8080
    - name: route-2
      match:
        hosts:
          - local.navendu.me
        paths:
          - /v2
      backends:
        - serviceName: bare-minimum-api-v2
          servicePort: 8081
```

这些 CRD 使配置 Ingress 变得更加容易，但您将绑定到特定的 Ingress 控制实现。如果 Ingress API 没有进化，您必须在可用性和可移植性之间进行选择。

扩展 Ingress 和 Gateway API 的演进

Ingress API 没有被打破；它受到了限制。Gateway API 的设计是为了克服这些限制。

> (Gateway API) 旨在通过表达性、可扩展和面向角色的接口演进 Kubernetes 服务网络...

它从前面提到的不同 Ingress 控制器的自定义 CRD 中汲取灵感。

Gateway API 在 Ingress API 的功能之上添加了 许多功能 。这包括基于 HTTP 标头的匹配、加权流量分割以及其他需要使用 Ingress API 的自定义专有注释的功能。

使用 APISIX Ingress 资源的流量分割 (请参阅 ApisixRoute/v2 reference):

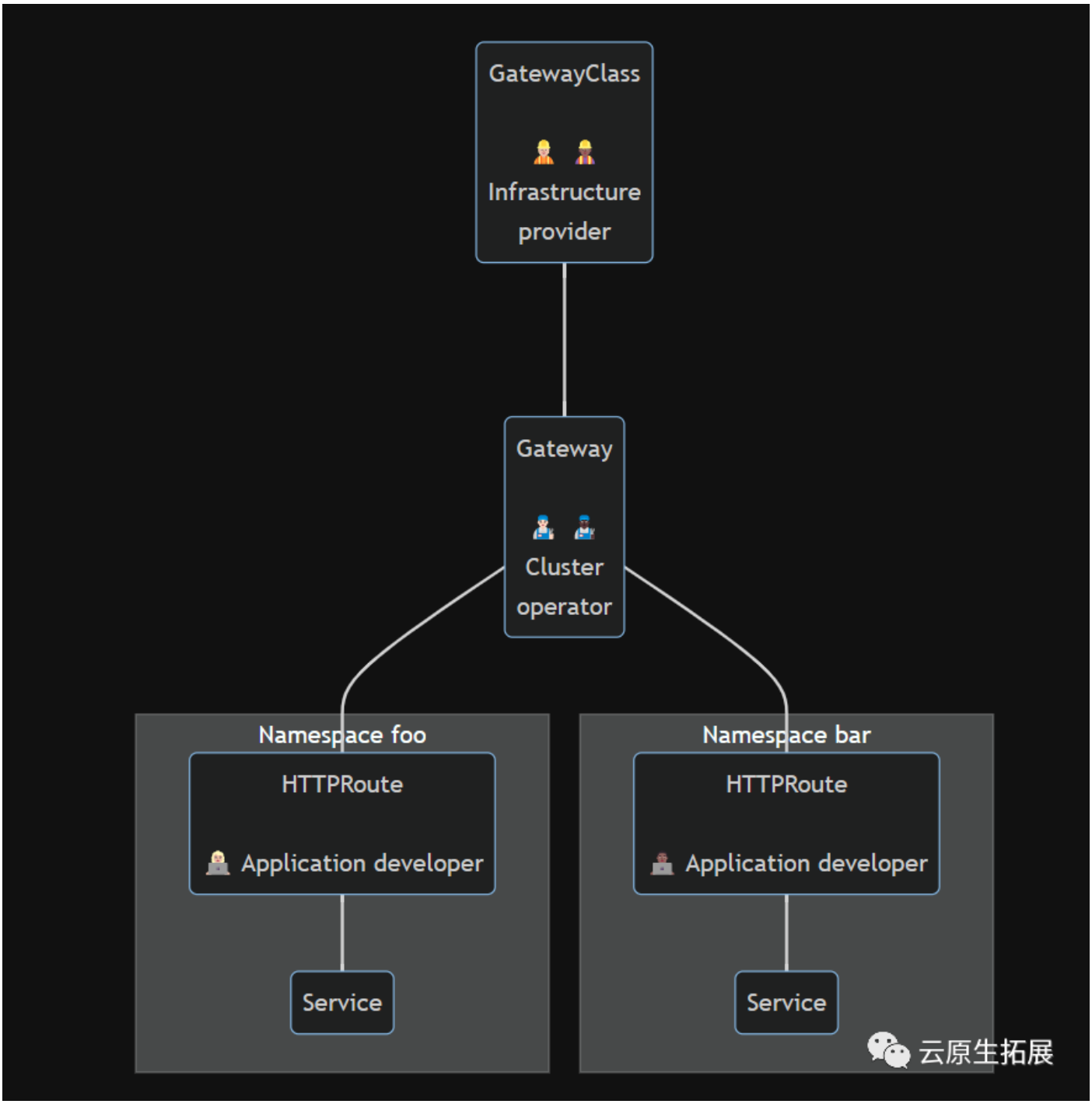
```
apisix-ingress-manifest.yaml
apiVersion: apisix.apache.org/v2
kind: ApisixRoute
metadata:
  name: traffic-split
spec:
  http:
    - name: rule-1
      match:
        hosts:
          - local.navendu.me
        paths:
          - /get*
      backends:
        - serviceName: bare-minimum-api-v1
          servicePort: 8080
          weight: 90
        - serviceName: bare-minimum-api-v2
          servicePort: 8081
          weight: 10
```

使用 Gateway API 的流量分割 (请参阅 金丝雀流量发布):

```
kubernetes-gateway-manifest.yaml
apiVersion: gateway.networking.k8s.io/v1alpha2
kind: HTTPRoute
metadata:
  name: traffic-split
spec:
  hostnames:
  - local.navendu.me
  rules:
  - backendRefs:
    - name: bare-minimum-api-v1
      port: 8080
      weight: 90
    - name: bare-minimum-api-v2
      port: 8081
      weight: 10
```

Ingress API 的另一个改进是 Gateway API 如何 分离关注点 。在 Ingress 中，应用程序开发人员和群集操作员在同一个 Ingress 对象上工作，不知道另一个人的责任，为误配置打开了大门。

Gateway API 将配置分解为 Route 和 Gateway 对象，为应用程序开发人员和群集操作员提供了自主权。下图清楚地解释了这一点:



Ingress API 终结?

Gateway API 是相对较新的，其实现不断地被打破。相比之下，Ingress API 处于稳定状态并经受住了时间的考验。

如果您的用例仅涉及简单路由，并且如果您可以使用自定义注释来获得额外的功能，则 Ingress API 仍然是一个稳定的选择。

由于 Gateway API 是 Ingress API 的超集，因此将两者合并可能是有意义的。感谢 SIG Network 社区，Gateway API 仍在发展，并将很快投入生产。

大多数 Ingress 控制器和 服务网格 已经实现了 Gateway API 和 Ingress API，随着项目的发展，将有更多的实现浮出水面。

个人而言，至少目前，我会坚持使用类似 Apache APISIX 的 Ingress 控制器提供的 自定义 CRDs ，而不是 Ingress 或 Gateway API。