DotNet 2021

ONLINE TECH CONFERENCE

#DotNet2021

22nd June 2021

Gateway API: El primo zumosol de Ingress

#KubernetesOnFire

www.dotnet2021.com

DotNet 2021

ORGANIZATION

plain concepts

IN COOPERATION WITH

FUNDACIÓN GOMAESPUMA

"Educando con una sonrisa."

SPONSORS

Microsoft

intelequia

Devs DNA™

My Public Inbox®

#DotNet2021

@eiximenis

etomas@plainconcepts.com

https://mypublicinbox.com/eiximenis

Eduard Tomàs

Rompiendo cosas en Plain Concepts

- Me pagan en Plain Concepts por aporrear un teclado (desconectado generalmente)
- Hago cerveza artesana, pero me la bebo
- Tengo un blog, en el que últimamente escribo con la misma frecuencia con la que George RR Martin saca libros nuevos.
  - https://eiximenis.dev
- Soy como Paco Umbral: Esa conferencia se llama **DotNet** pero yo voy a hablar de Kubernetes.

Ingress

Expongo tus servicios en Kubernetes de forma pública

Gateway API

Soy el primo Zumosol de Ingress

Los problemas de Ingress

- **Anotaciones, anotaciones, anotaciones**
- Se usan anotaciones para especificar opciones (necesarias) pero que no existen en la spec
- 301 a HTTPs, tamaños de headers, CORS, versiones de TLS, etc, etc, etc, etc, etc, etc, etc,…

Los problemas de Ingress

- **Un solo servicio por ruta**
- El servicio puede abarcar pods de distintas versiones de la app
- Pero p. ej. para redirigir el 99% del tráfico a v1 y el 1% a v2, **necesitamos 99 pods de v1 corriendo por cada pod de v2**

Los problemas de ingress

Un mismo host virtual puede estar definido en distintos objetos ingress

Estos objetos ingress pueden estar en namespaces distintos

Eso permite a equipos distintos (con accesos a namespaces distintos) configurar rutas sobre el mismo virtual host ☺

Pero no hay forma de evitar precisamente eso: cualquiera que tenga permisos para publicar un ingress en cualquier namespace, puede inyectar rutas en cualquier host virtual ☹

Además, es complicado que una futura versión de Kubernetes aborde esa cuestión porqué precisamente proyectos como cert-manager dependen de eso para poder usar el challenge HTTP-01 de ACME al generar certificados TLS ☹

Los problemas de ingress

Default backend… ese gran desconocido

Gestiona las peticiones que no se cumplen ninguna de las reglas de ingress

El problema es que puede (suele) haber varios ingress, cada uno puede especificar un default backend distinto…

En este caso, ¿cual debe usarse? O deben combinarse? Pero como?

Esa ambigüedad parece “sugerir” que el default backend puede ser un objeto de ámbito de namespace asociado a su ingress, pero… no

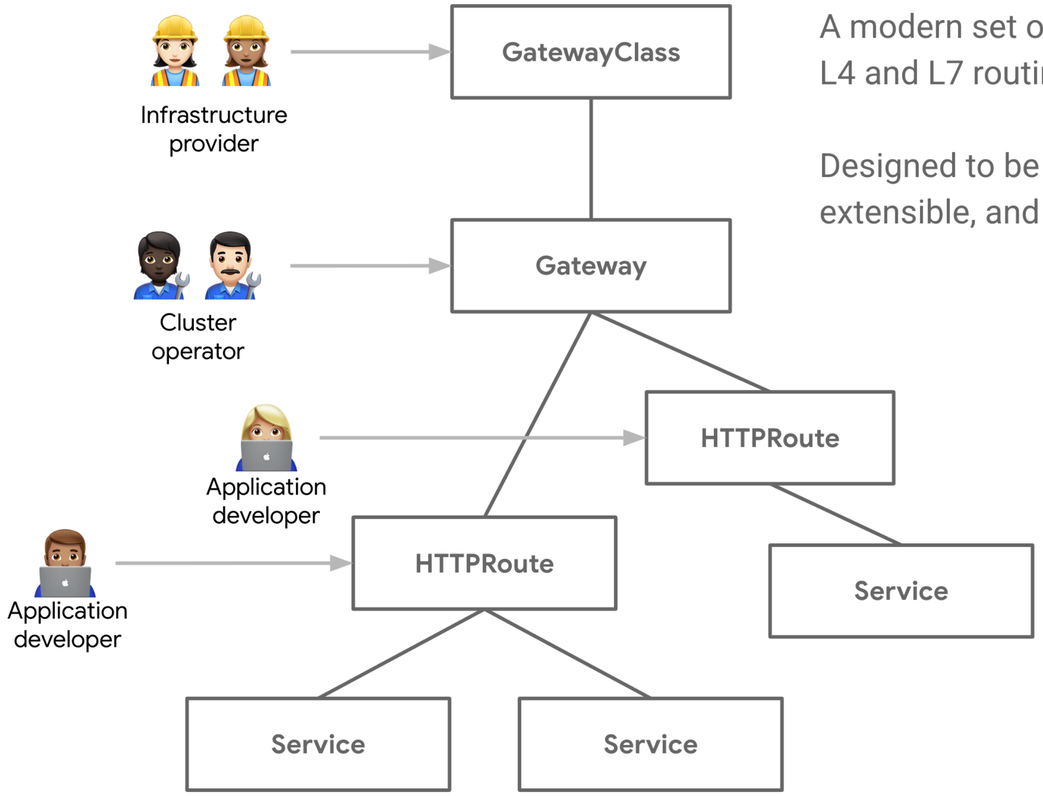
Kubernetes Gateway API

Separa claramente los roles del **operador del clúster de los del desarrollador de la aplicación**

Operador: Define que Gateways tiene el clúster instalados

Desarrollador: Define las rutas de sus aplicaciones

Requiere un controlador de API Gateway para funcionar



A modern set of APIs for deploying L4 and L7 routing in Kubernetes

Designed to be generic, expressive, extensible, and role-oriented

Conceptos

- **GatewayClass:** Recurso a nivel de cúster que define el proveedor de infraestructura. Define que tipo de Gateways se pueden instanciar.
  - Juega un rol similar al de IngressClass ya que permite enlazar Gateways con controladores instalados y parametrizar estos

- **Gateway**: Representa un balanceador de carga operando fuera del clúster. Expone puertos así como define protocolos pero **a diferencia** de un recurso Ingress **no contiene regla de enrutado alguna**.
  - Al desplegar un Gateway es cuando se configura y/o aprovisiona la infraestructura necesaria a través del controlador de API Gateway asociado a través de la GatewayClass

- **HTTPRoute**: Representa una ruta HTTP/HTTPS que puede ser usada por una o más Gateways

Gateway API vs Ingress

Al **igual que Ingress** se trata de una **especificación de API**. Se confía en los controladores para que la implementen

Más expresiva que Ingress: al estar basada en varios CRDs es posible especificar opciones de extension que en ingress solo existían a través de las anotaciones

Es más extensible que Ingress: Soportar protocolos específicos (p. ej. gRPC) es posible a través de tipos de rutas adicionales

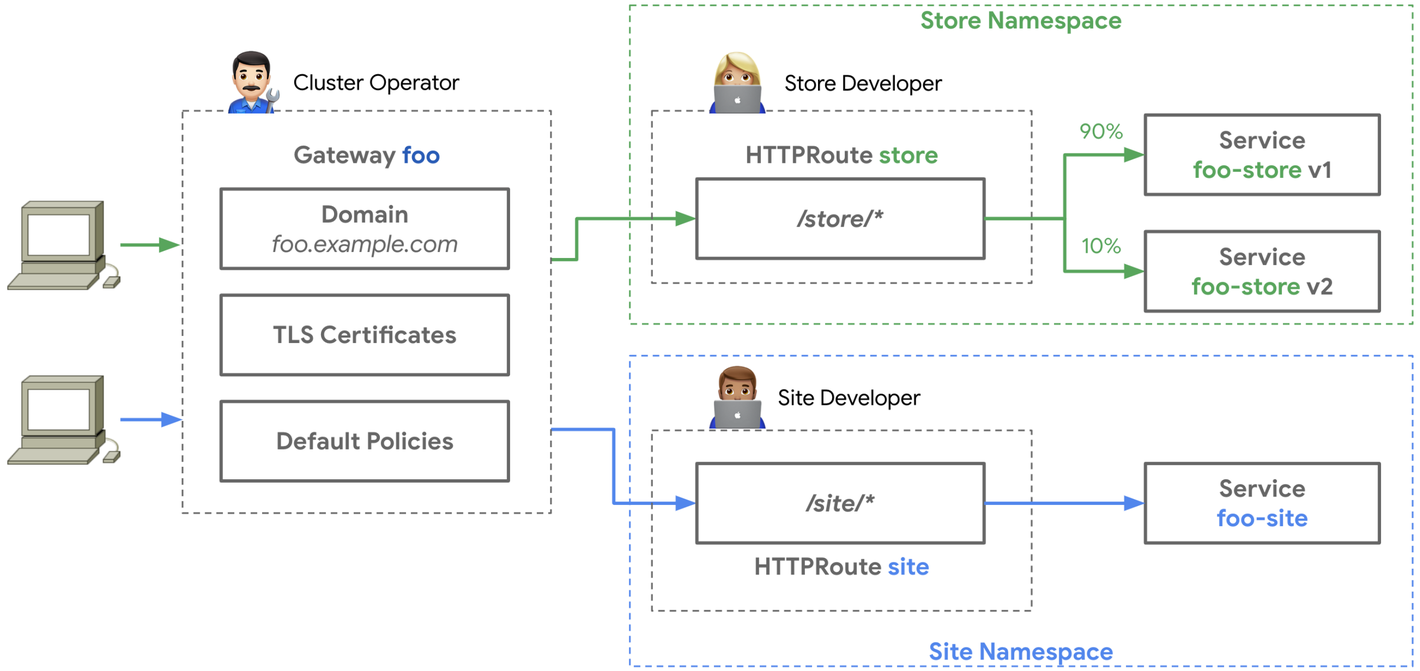
Gateway API: Orientada a roles

El **operador de cluster define las Gateways** existentes en el cluster

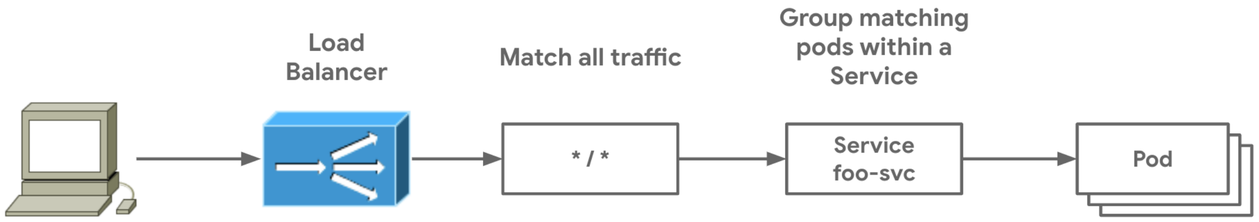
El **operador de cluster** define qué rutas (p. ej en qué namespaces) pueden añadirse a la Gateway y de ese modo exponer aplicaciones a través de ésta

El **operador de cluster** puede forzar políticas de TLS que no puedan ser redefinidas luego

Los **desarrolladores** de aplicaciones, se limitan a decidir las rutas, protocolos y condiciones necesarias para sus aplicaciones



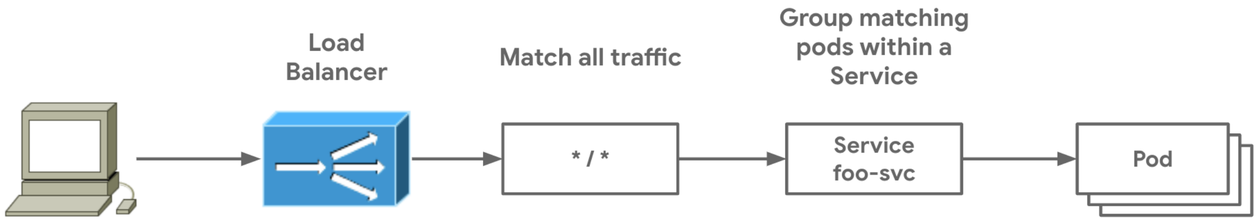
Escenario 1: Ingress Like



- El *Gateway* representa un balanceador de carga específico
- Este *Gateway* escuchará todo el tráfico HTTP por el puerto 80
- Y lo enviará a sus rutas seleccionadas
- El *Gateway* selecciona sus rutas en base a un selector (app:kuard)

```yaml
kind: Gateway
apiVersion: networking.x-k8s.io/v1alpha1
metadata:
  name: simple
  namespace: demo1
spec:
  gatewayClassName: sample-gatewayclass
  listeners:
    - protocol: HTTP
      port: 80
      routes:
        kind: HTTPRoute
        selector:
          matchLabels:
            app: kuard
```

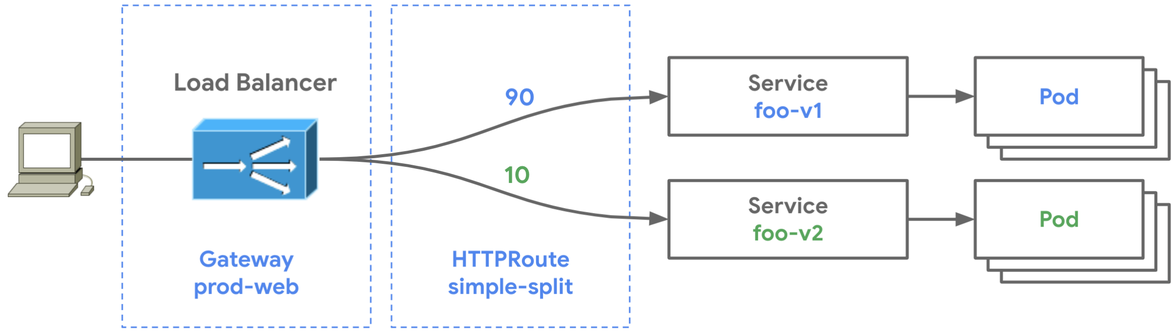
Escenario 1: Ingress Like



- HTTPRoute sirve para enrutar peticiones HTTP/HTTPS
- Se parece a una ruta de ingress: asocia hostnames a reglas, que se aplican para enrutar el tráfico al servicio indicado
- Las rutas quedan asociadas al Gateway que las sirve en base a sus etiquetas

```yaml
kind: HTTPRoute
apiVersion: networking.x-k8s.io/v1alpha1
metadata:
  name: kuard
  namespace: demo
  labels:
    app: kuard
spec:
  hostnames:
    - "demo1.localtest.me"
  rules:
    - matches:
        - path:
            type: Prefix
            value: /
      forwardTo:
        - serviceName: kuard
          port: 80
```

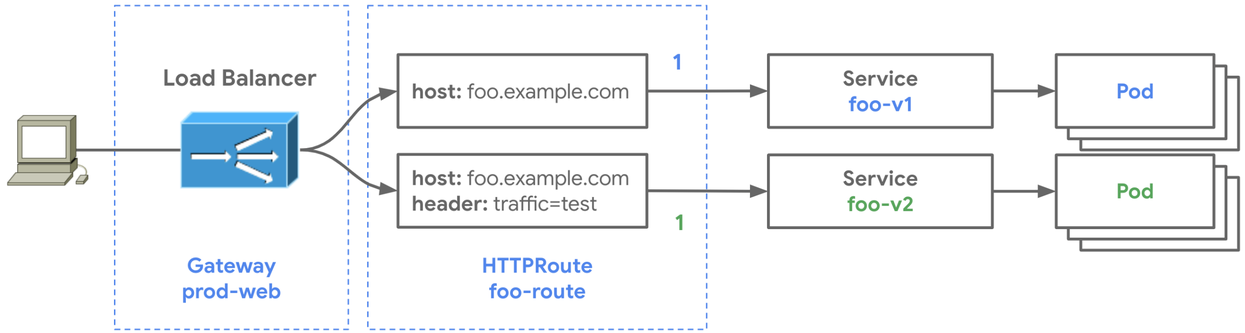
Escenario 2: Traffic Splitting



- Da soporte a escenarios como:
  - Pruebas A/B
  - Blue/Green deployment
  - Canary releases

```yaml
kind: HTTPRoute
apiVersion: networking.x-k8s.io/v1alpha1
metadata:
  name: web
  namespace: demo
  labels:
    app: dotnet2021
spec:
  hostnames:
    - "demoweb.localtest.me"
  rules:
    - matches:
        - path:
            type: Prefix
            value: /
      forwardTo:
        - serviceName: web1
          port: 80
          weight: 50
        - serviceName: web2
          port: 80
          weight: 50
```

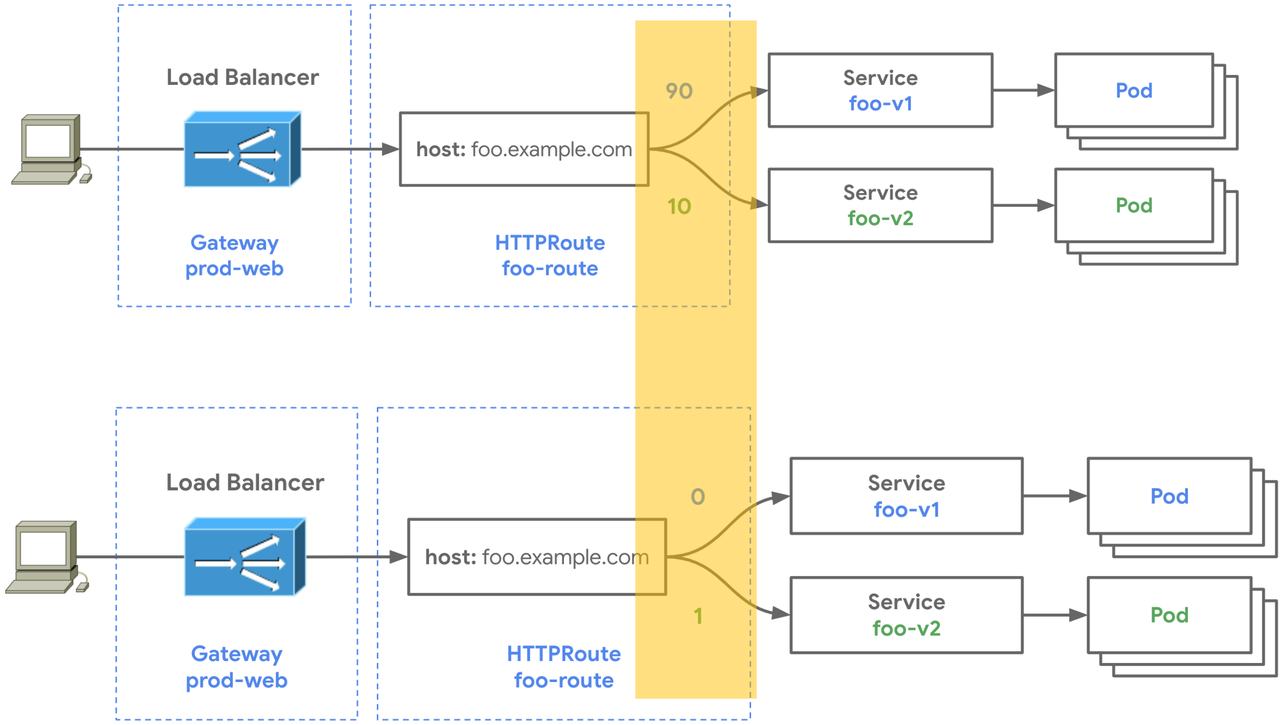
Escenario 2b: Canary release



- Da soporte a escenarios como:
  - Pruebas A/B
  - Blue/Green deployment
  - Canary releases

```yaml
kind: HTTPRoute
apiVersion: networking.x-k8s.io/v1alpha1
metadata:
  name: canary
  namespace: demo
  labels:
    app: dotnet2021
spec:
  hostnames:
    - "democanary.localtest.me"
  rules:
  - forwardTo:
    - serviceName: web1
      port: 80
  - matches:
    - headers:
        type: Exact
        values:
          traffic: test
    forwardTo:
      - serviceName: web2
        port: 80
```

Escenario 2c: Blue/Green deploy




```yaml
kind: HTTPRoute
apiVersion: networking.x-k8s.io/v1alpha1
metadata:
  name: web
  namespace: demo
  labels:
    app: dotnet2021
spec:
  hostnames:
    - "demoweb.localtest.me"
  rules:
    - matches:
        - path:
            type: Prefix
            value: /
      forwardTo:
        - serviceName: web1
          port: 80
          weight: 0
        - serviceName: web2
          port: 80
          weight: 100

.
weight: 50
```

Mapeo de TCP

A diferencia de ingress, API Gateway soporta otros protocolos además de HTTP/HTTPS

Se pueden exponer servicios con protocolos TCP propios (bases de datos, sistemas de mensajería, etc) a través de una Gateway

Esa Gateway tiene los puertos 8080 y 8090 listos para enrutar peticiones TCP.

Para el puerto 8080 usará TCPRoutes con la etiqueta “app” a “foo”

Para el puerto 8090 usará TCPRoutes con la etiqueta “app” a “bar”

```yaml
kind: Gateway
apiVersion: networking.x-k8s.io/v1alpha1
metadata:
  name: my-tcp-gateway
spec:
  gatewayClassName: acme-lb
  listeners:
  - protocol: TCP
    port: 8080
    routes:
      kind: TCPRoute
      selector:
        matchLabels:
          "app": "foo"
  - protocol: TCP
    port: 8090
    routes:
      kind: TCPRoute
      selector:
        matchLabels:
          "app": "bar"
```

Mapeo de TCP

Esa ruta TCP reenvía el tráfico al servicio sqlserver al puerto 1433

Como tiene la etiqueta “app=foo” esa ruta es seleccionada por la API Gateway a través del puerto 8080

Por lo tanto acceder al cluster usando el puerto 8080 redireccionará el tráfico al SQL Server

```yaml
kind: TCPRoute
apiVersion: networking.x-k8s.io/v1alpha1
metadata:
  name: tcp-app-1
  labels:
    app: foo
spec:
  rules:
  - forwardTo:
    - serviceName: sqlserver
      port: 1433
```

Soporte para varios namesapces

Una *Gateway* puede tener rutas asociadas que estén en otros espacios de nombres. Para ello:

1. La *Gateway* debe indicar que *puede* seleccionar rutas de todos los espacios de nombres

2. La ruta debe cumplir el selector **y además debe especificar en spec.gateways.allow** que permite que gateways de otros espacios de nombres usen la ruta

TLS

La configuración TLS de downstream se configura en la Gateway, siguiendo un modelo parecido al de Ingress

Se define un secreto que contiene el certificado que la Gateway debe servir en cada caso al cliente

```
listeners:
- protocol: HTTPS # o TLS
  port: 443
  tls:
    mode: Terminate # o Passthrough
    certificateRef:
      kind: Secret
      group: core
      name: default-cert
    routeOverride:
      certificate: Deny
```

TLS

La configuración TLS de upstream se configura usando un recurso nuevo, llamado *BackendPolicy*

Esa configuración es relativa **al servicio**, no a la ruta

```yaml
kind: BackendPolicy
apiVersion: networking.x-k8s.io/v1alpha1
metadata:
  name: my-app
  annotations:
    networking.x-k8s.io/app-protocol: https
spec:
  backendRefs:
  - name: my-service
    group: core
    kind: Service
    port: 443
  tls:
    certificateAuthorityRef:
      name: my-cluster-ca
      group: core
      kind: Secret
    options: {}
```

Bueno, vale, pero…

Todo eso… se va a cargar a Ingress?

No, ingress es una API GA de Kubernetes que no se espera que sea declarada obsoleta por el momento. Ingress no está rota, lo que si tiene sus límites que es lo que se intenta solucionar con esa API Gateway

Cuan maduro está eso?

A día de hoy (22/06/2021) todo esto está en alfa y las implementaciones de controladores de API Gateway están justo empezando a salir. Aunque puedes empezar a jugar con ellos ten presente que **como todas las APIs alfa en Kubernetes, seguramente se sufran varios cambios**, antes de que esa API quede fijada. Así que todo lo dicho en esta charla puede cambiar ☺

DotNet 2021

ONLINE TECH CONFERENCE

www.dotnet2021.com

#DotNet2021

Thanks and ...
See you soon!

Thanks also to the sponsors. Without whom this would not have been posible.