

The Future of Ingress in Kubernetes



Nigel Brown

@n_brownuk www.windsock.io



Module Outline



Explore the limitations of the Ingress API

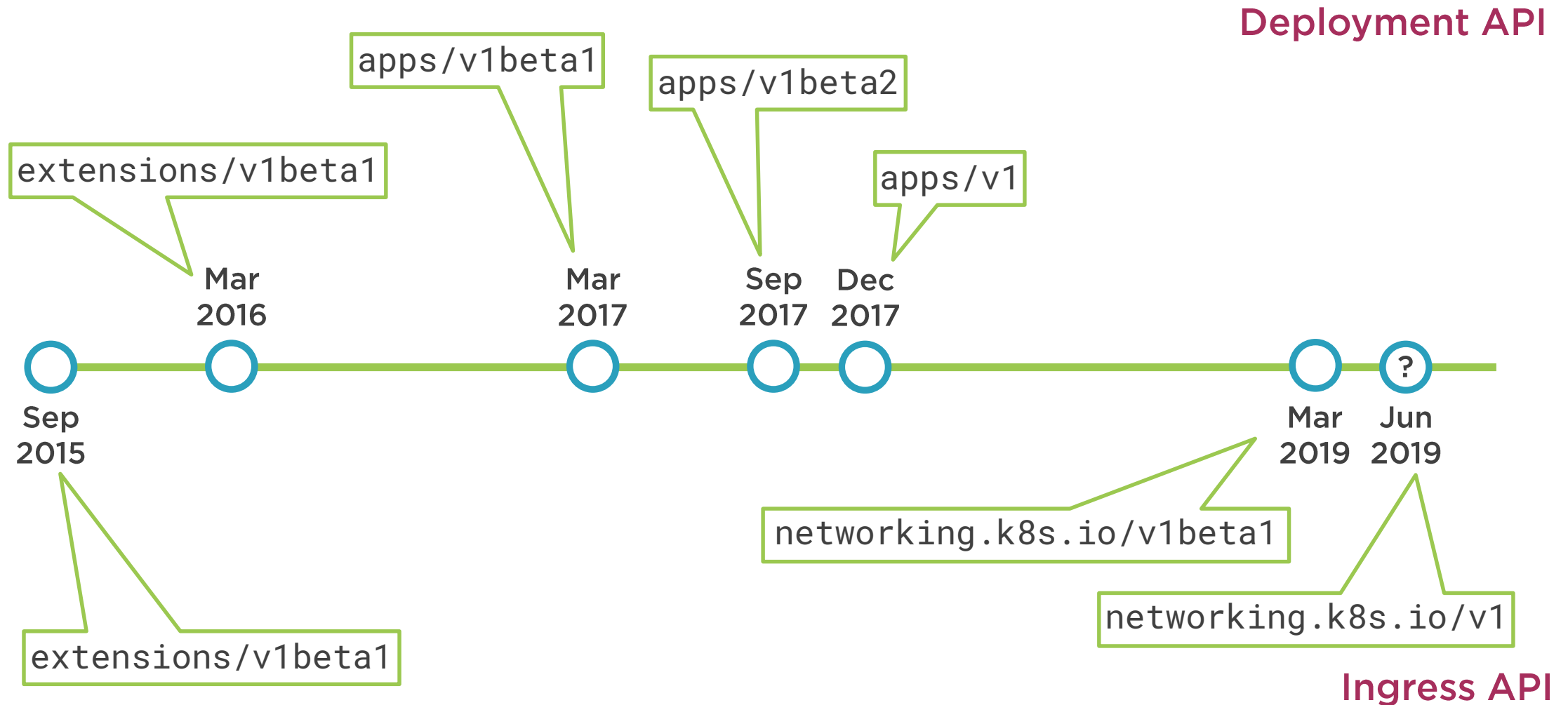
Look at an enhanced ingress-related API called IngressRoute

Explain how to avoid the Ingress API altogether using an API gateway

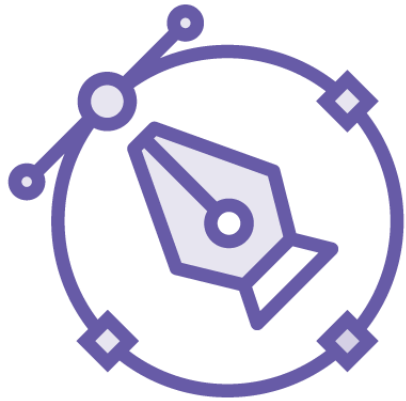
Discuss ingress in the context of a service mesh



Slow Evolution of the Ingress API



Compromise Solution with Side Effects



Limited by Design

Ingress API is deliberately terse



Annotation Stopgap

Provided as method to overcome deficiency

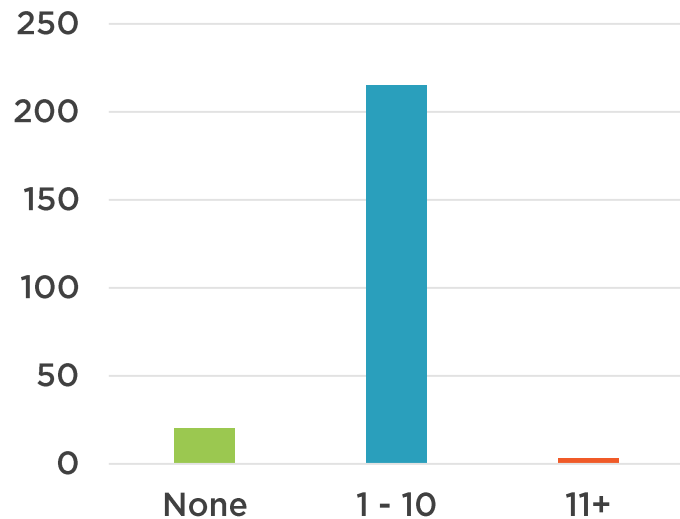


Impractical Solution

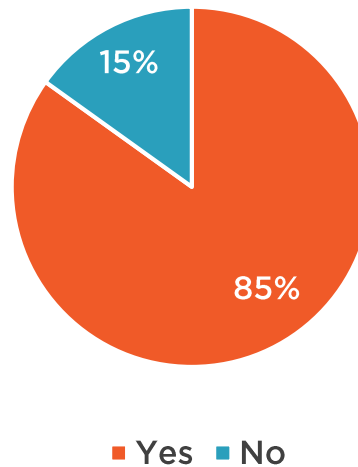
Explosion of disparate annotation patterns

Kubernetes Ingress Survey 2018

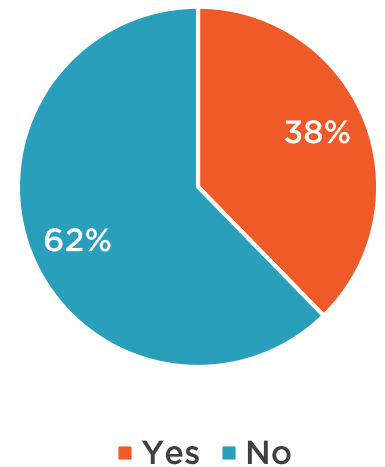
Annotations per
ingress definition?



Ingress definitions
should be portable?



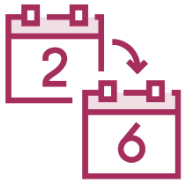
Is Ingress API
expressive enough?



<https://git.io/fjT7c>



Future Direction of Ingress



Network SIG is likely to facilitate a consensus for a future v2 Ingress API or equivalent



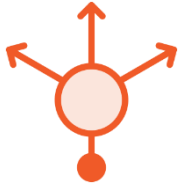
Some ingress controllers types are starting to use custom resource definitions (CRD)



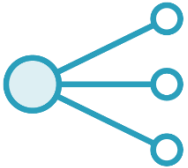
Ingress controller users need to select the tool that best suits the intended purpose



Contour Ingress Controller



Contour ingress controller provides a richer experience for ingress traffic configuration, without annotations



The basic Kubernetes Ingress API is still supported by Contour, along with a number of annotations



Contour comes with a Custom Resource Definition (CRD) called **IngressRoute** for enhanced ingress use



```
apiVersion: contour.heptio.com/v1beta1
kind: IngressRoute
metadata:
  name: nginxhello-ingress
```

Defining an IngressRoute Object

Contour's CRD has an API kind called IngressRoute




```
spec:
  virtualhost:
    fqdn: dibble.sh
  routes:
    - match: /
      services:
        - name: nginxhello
          port: 80
```

IngressRoute Spec Basics

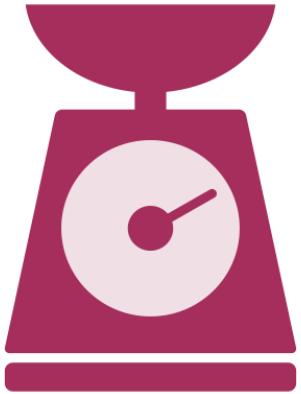
A separate IngressRoute definition is required per virtual host

Fully qualified domain name is matched against request's Host header

Routes comprise matched paths and one or more service backends

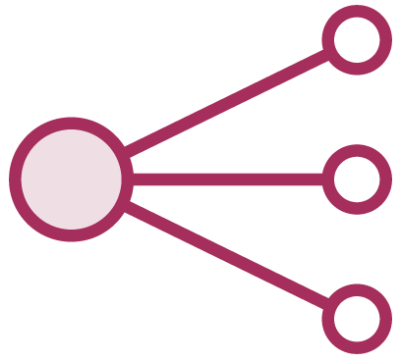


IngressRoute CRD Capabilities



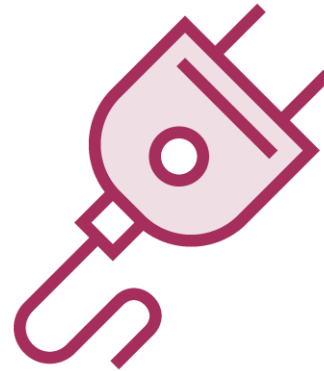
Service Weights

Traffic routed by
service weights



Load Balancing

Different service
LB algorithms



WebSockets

Connections can
be upgraded



Path Rewrite

URL paths can
be prefixed

IngressRoute Delegation

```
apiVersion: contour.heptio.com/v1beta1
kind: IngressRoute
metadata:
  name: dibble-sh
  namespace: admin
spec:
  virtualhost:
    fqdn: dibble.sh
  routes:
    - match: /
      services:
        - name: nginxhello
          port: 80
    - match: /blue
      delegate:
        name: blue-dibble-sh
        namespace: blue
```



A diagram illustrating IngressRoute delegation. An orange oval highlights the 'delegate' field in the 'match: /blue' route of the first IngressRoute (admin namespace). A line connects this oval to another orange oval in the second IngressRoute, which highlights the 'name: blue-dibble-sh' and 'namespace: blue' fields under the 'spec.routes' section.

```
apiVersion: contour.heptio.com/v1beta1
kind: IngressRoute
metadata:
  name: blue-dibble-sh
  namespace: blue
spec:
  routes:
    - match: /blue
      services:
        - name: nginxhello-blue
          port: 80
          strategy: RoundRobin
```

TLS delegation, TCP Proxying ...
<https://git.io/fjkiW>





Ingress and IngressRoute

In the future, IngressRoute may well form the basis of a revised Ingress API



Ambassador Microservices API Gateway



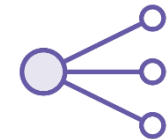
Ambassador is an open source API gateway designed for use in Kubernetes clusters



Uses Envoy under the covers to provide microservices API gateway and ingress functions



Bypasses the Kubernetes Ingress API, and configures Envoy using annotations on Service objects



Ambassador will ordinarily be fronted by a Service object of type LoadBalancer



Ambassador Capabilities

Routing by
Header

URL Path Prefix
Rewrites

Canary Release
Deployments

Traffic Shadowing
Deployments

Addition of
Headers

TCP Traffic
Proxying



```
apiVersion: v1
kind: Service
metadata:
  name: nginxhello
  labels:
    app: nginxhello
  annotations:
    getambassador.io/config: |
      ---
      apiVersion: ambassador/v1
      kind: Mapping
      name: nginxhello-mapping
      host: dibble.sh
      prefix: /
      service: nginxhello 80
spec:
```

- ◀ Standard k8s service definition
- ◀ Ambassador-specific annotation
- ◀ Version of the Ambassador API, resource kind and name
- ◀ Host header and path
- ◀ Target service

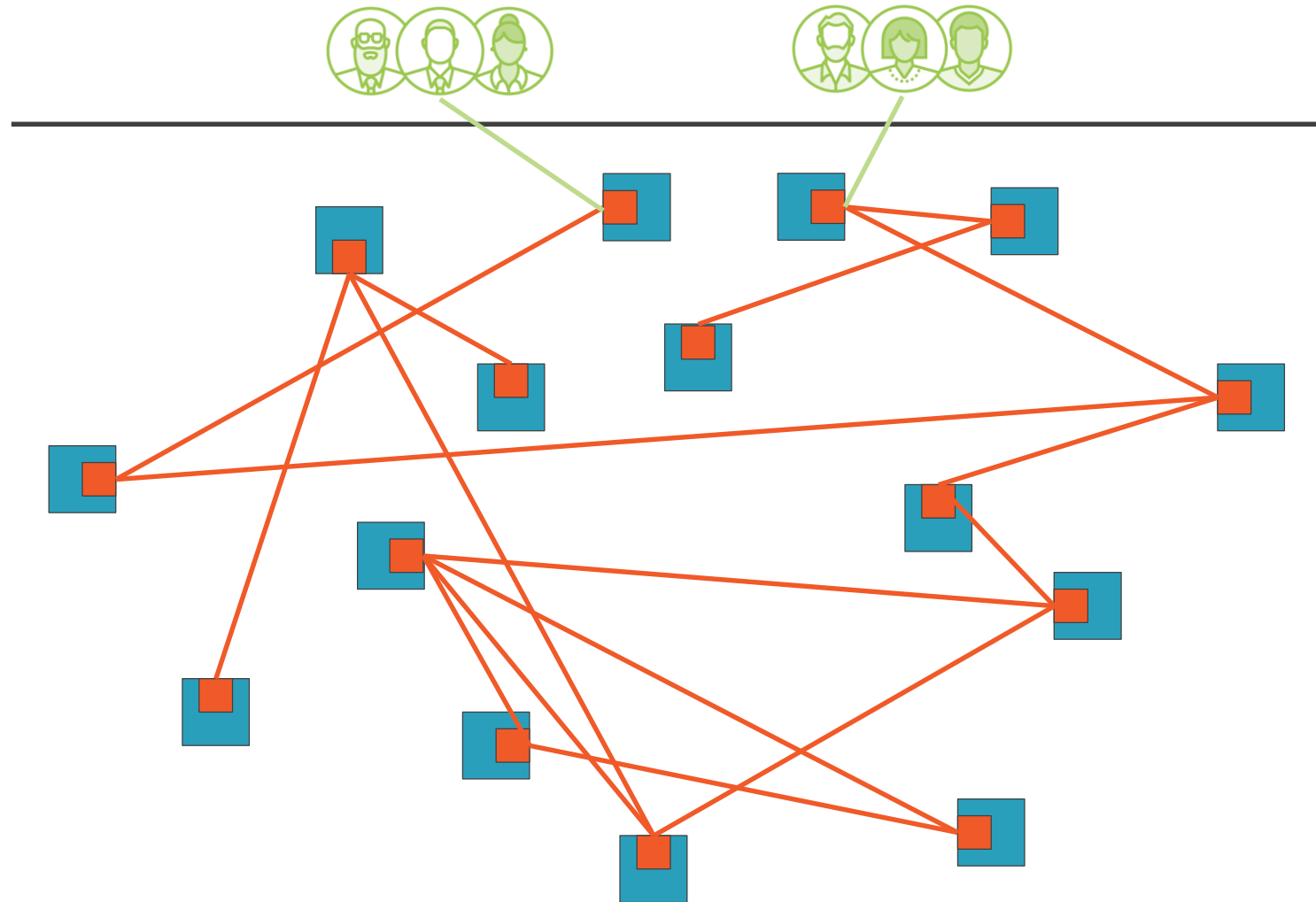


Istio

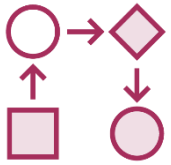
Istio is a platform for managing the complexity of microservices deployments, using a service mesh abstraction.



Service Mesh



Istio Ingress Gateway



Ingress is performed by a suitably configured deployment of the Envoy proxy



Gateway objects define the types of ingress traffic allowed to enter the mesh



Rules for routing traffic to services in the mesh, are defined using VirtualService objects



Traffic policy is defined and applied to routes using DestinationRule objects



Ingress Options

Ingress API

Contour IngressRoute

Ambassador API Gateway

Istio Ingress Gateway



Course Summary



Thank you!

Kubernetes Service objects provide a simple ingress option

The Ingress API is the purpose built feature for handling ingress

Cert-manager builds on the Ingress API for TLS certificate management

Ingress controllers provide advanced features accessible through annotations

Other community-led initiatives provide a richer ingress experience



Where to Go Next

Kubernetes Ingress	https://k8s.io/docs/concepts/services-networking/ingress/
Nginx Ingress Controller	https://git.io/fjLR4
Traefik Ingress Controller	https://docs.traefik.io/user-guide/kubernetes/
Contour Ingress Controller	https://git.io/fh4Ua
cert-manager	https://docs.cert-manager.io/en/latest/
Ambassador API Gateway	https://www.getambassador.io/docs
Istio Service Mesh	https://istio.io/

