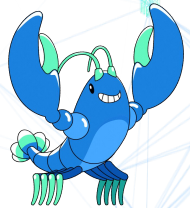
BUOYANT

Creators of LINKERD



Gateway API 101 with Linkerd



@BuoyantIO

buoyant.io

SERVICE MESH ACADEMY



Shane Utt
Senior Principal Software Engineer
Red Hat



Mattia Lavacca
Software Engineer
Kong



Flynn
Tech Evangelist
Buoyant

gateway api

SERVICEMESH ACADEMY



Shane Utt
SIG Network Chair
Gateway API Maintainer



Mattia Lavacca
Gateway API
Maintainer



Flynn
Gateway API
Mesh Co-lead

gateway api



What's on the agenda?

- What is Gateway API?
  - and why you should care
- DEMO!
- Gotchas

gateway api

BUOYANT

How do you follow along?



- https://github.com/BuoyantIO/service-mesh-academy/tree/main/gateway-api-101
- **For this demo, I'll be using Linkerd enterprise-2.17.0**
  - You'll need a free Buoyant ID from https://enterprise.buoyant.io/
- Everything I'm demoing will also work with edge-25.3.1 or later

gateway api

BUOYANT

How do you follow along?



- I'll be using a k3d cluster, but really any cluster should work fine...
- ...as long as you have working LoadBalancer services.

How do you follow along?



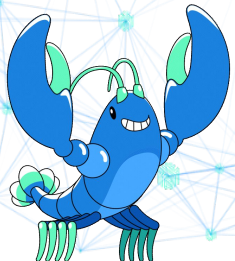
- **kubectl**
  https://kubernetes.io/docs/tasks/tools/
- **k3d**
  https://k3d.io/
- **bat**
  https://github.com/sharkdp/bat
- **helm**
  https://helm.sh/docs/intro/quickstart



BUOYANT
SERVICE MESH
ACADEMY

gateway api

What is Gateway API?





Gateway API is...

- A project within Kubernetes SIG-Network
- Started in 2019 (Kubecon San Diego) as “Ingress v2”
    - and a potential successor to “Service” as well
- Reached GA in October 2023
- Stable, but definitely **not** done by any means!

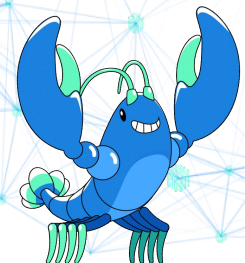




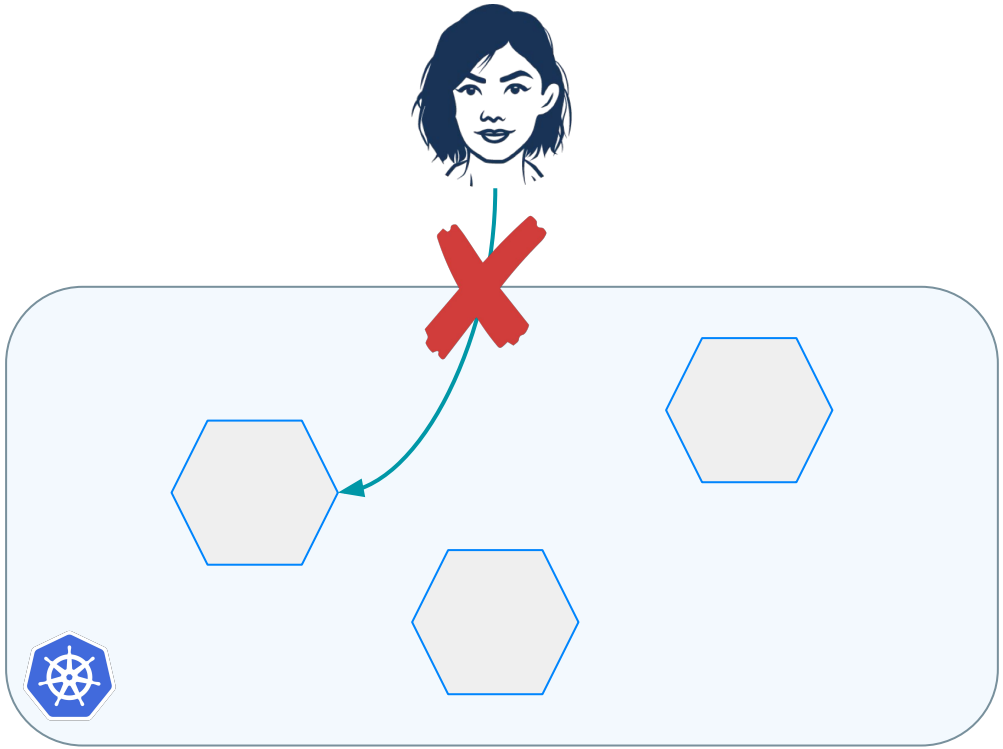
BUOYANT
SERVICE MESH
ACADEMY

gateway api

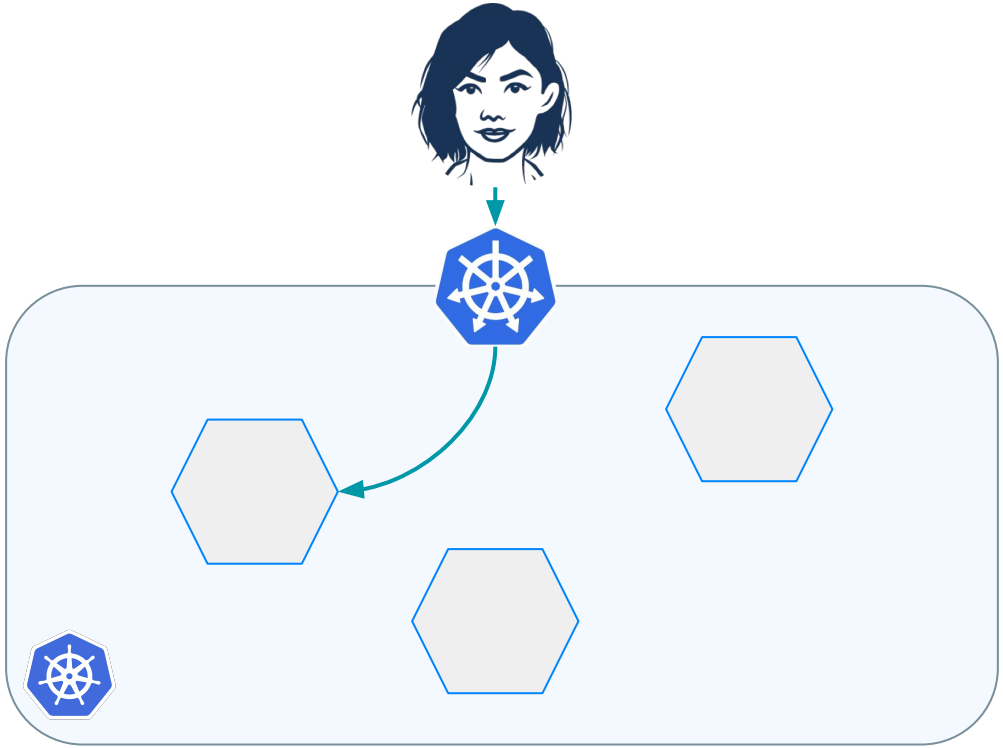
The Ingress Problem



How do you let people use your services?



How do you let people use your services?





Back in the bad old days...

We had the Ingress resource.

- (Still) widely used, but it has problems.
- Monolithic - one resource for everything, no separation of concerns
- Limited scope - many basic features weren't covered
- Limited extensibility - no formal extension mechanism led to the Wild West of annotations (more on this later)

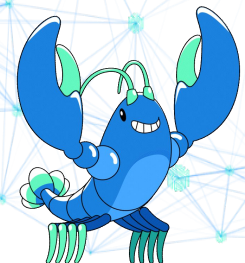




BUOYANT
SERVICE MESH
ACADEMY

gateway api

From Ingress to Gateway API





Gateway API is an attempt to fix this

What would be better?

- Standardization for common capabilities
- Expanded scope - eventually: today work continues to cover the basics
- Design for extensibility - have standardized extension mechanisms
- Role-oriented - last but definitely not least, separate concerns based on real-world needs





Gateway API is a standard

- Learn how a concept works, use it with whatever infrastructure you have
- Things that are part of the standard are widely supported
  - Things that are less widely supported can be standardized as “Extended” behaviors
  - In general, nothing gets standardized without several implementations





Gateway API will cover the basics

- Today: HTTP/GRPC routing
  - basic stuff: “route this URL path to this Service”
  - advanced stuff: “route this URL path to Service A if it has the `X-Foo: Bar` header, otherwise route 25% of the traffic to Service B and the rest to Service C”
  - TCP/UDP/TLS exists, but remains in Experimental status
- Currently still several critical behaviors that aren’t yet standard
  - For example, CORS is just becoming standard
- Iterations continue, releases multiple times per year





Gateway API can be extended

- Gateway API was designed to allow extension where necessary
  - Implementations can support *more* than the standard resources
  - Some resources include specific “ExtensionRef” fields where implementation-specific resources can be referenced
  - If all else fails, there’s policy attachment
    - (but if you can avoid policy attachment, do so)



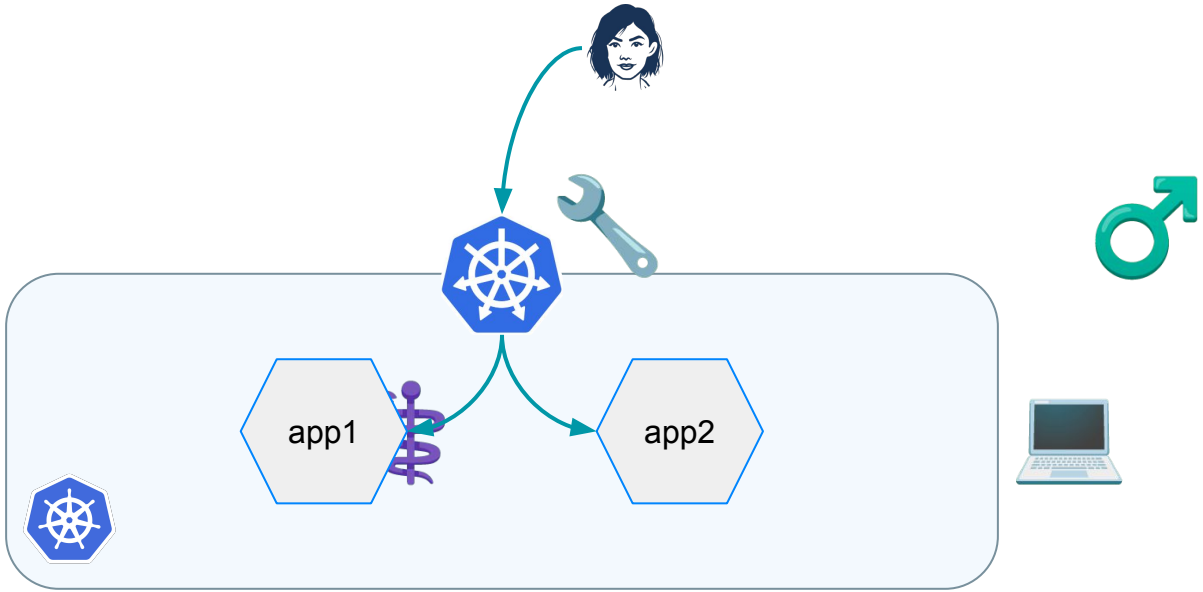
BUOYANT
SERVICE MESH
ACADEMY

gateway api

Role Oriented API Design



Gateway API is role-oriented

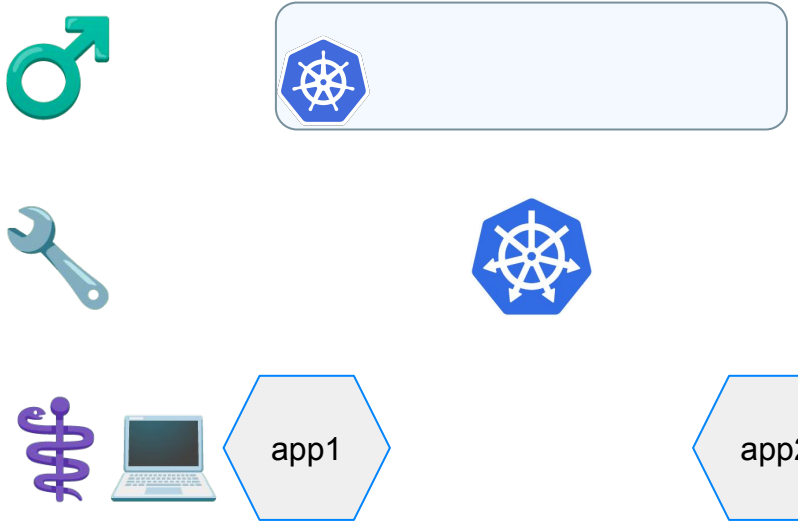


Gateway API is role-oriented

Ian
infrastructure provider

Chihiro
cluster operator

Ana
application developer



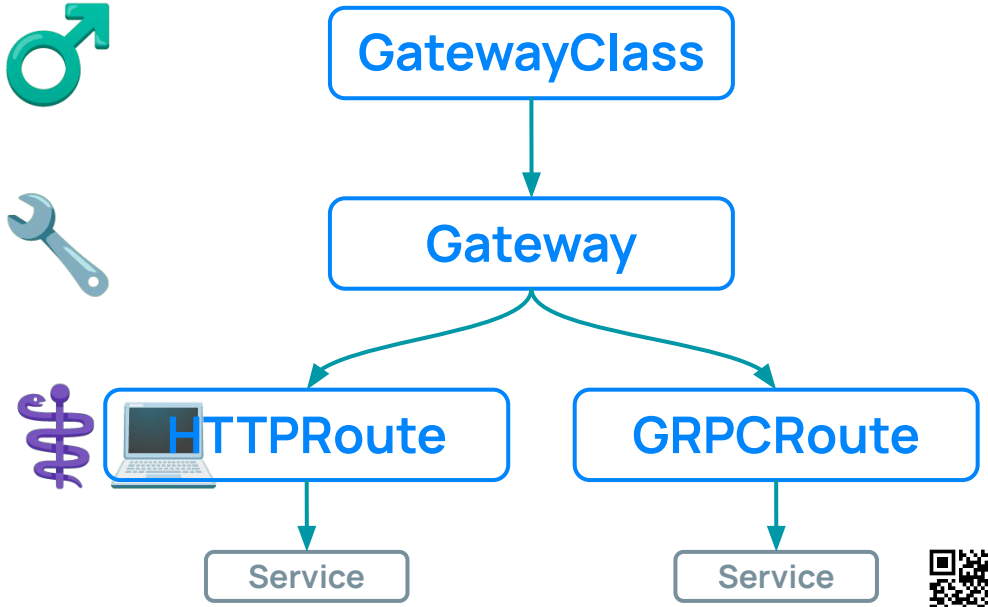


Gateway API is role-oriented

Ian
infrastructure provider

Chihiro
cluster operator

Ana
application developer



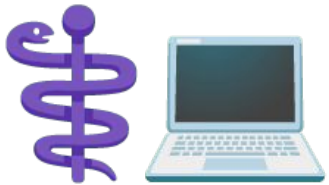




Gateway API has one most important role

Ana

application developer

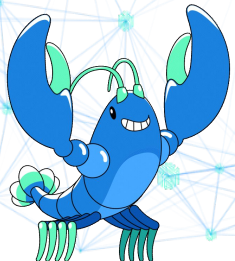




BUOYANT
SERVICE MESH
ACADEMY

gateway api

The Resources





HTTPRoute 👩‍💻

HTTPRoute defines how to route HTTP

- HTTP traffic arriving at the `faces-gateway`…
- with a `Host` header of `smiley.example.com`…
- gets routed to the `smiley` Service on port 80

```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: simplest-smiley-route
  namespace: faces
spec:
  parentRefs:
    - kind: Gateway
      name: faces-gateway
  hostnames:
    - smiley.example.com
  rules:
    - backendRefs:
      - name: smiley
        port: 80
```





HTTPRoute 👩‍💻

- We can do a 50/50 split across `smiley` and `smiley2` if we want.

```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: simplest-smiley-route
  namespace: faces
spec:
  parentRefs:
    - kind: Gateway
      name: faces-gateway
  hostnames:
    - smiley.example.com
  rules:
    - backendRefs:
      - name: smiley
        port: 80
      - name: smiley2
        port: 80
```





HTTPRoute 👩‍💻

- Or add a `Path` match.

```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: simplest-smiley-route
  namespace: faces
spec:
  parentRefs:
    - kind: Gateway
      name: faces-gateway
  hostnames:
    - smiley.example.com
  rules:
    - matches:
      - path:
        - value: /good/path
    - backendRefs:
      - name: smiley
        port: 80
      - name: smiley2
        port: 80
```





GRPCRoute 👩‍💻

- GRPCRoute is much the same, just optimized for gRPC.

```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: GRPCRoute
metadata:
  name: color-edge-route
  namespace: faces
spec:
  parentRefs:
    - kind: Gateway
      name: faces-gateway
  hostnames:
    - color.example.com
  rules:
    - matches:
      - method:
          service: ColorService
          method: Edge
    - backendRefs:
      - name: color
        port: 80
      - name: color2
        port: 80
```





Gateway 👷

Gateway, by contrast, focuses on totally different things.

- like what kind of gateway to make
- where it should listen
- and which routes it should accept

```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: ingress-gateway
  namespace: default
spec:
  gatewayClassName: ingress-gateway-class
  listeners:
    - name: http
      protocol: HTTP
      port: 80
      allowedRoutes:
        namespaces:
          from: all
```





GatewayClass 👷

GatewayClass is really a rendezvous and extension point for Gateways

- Gateway controllers look for GatewayClasses with their `controllerName`…
- …then accept Gateways with matching `gatewayClassName`s.
- Finally, `parametersRef` is for configuration or extensions

```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: GatewayClass
metadata:
  name: ingress-gateway-class
spec:
  controllerName: my-awesome-controller
  parametersRef: ...
```

```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: ingress-gateway
  namespace: default
spec:
  gatewayClassName: ingress-gateway-class
  listeners: ...
```



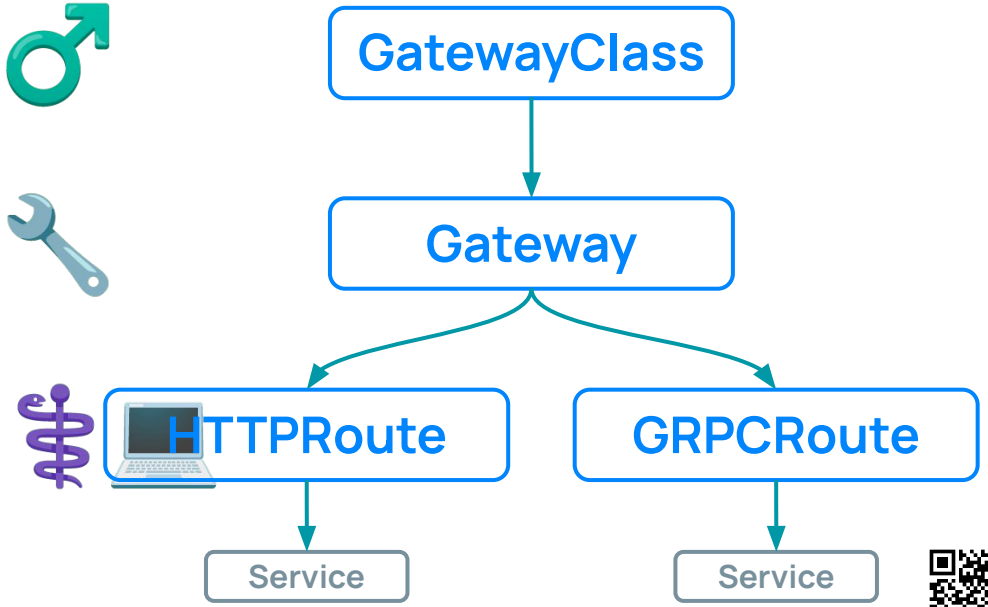


Gateway API is role-oriented

Ian
infrastructure provider

Chihiro
cluster operator

Ana
application developer







ReferenceGrant 👷

ReferenceGrant is a way to explicitly permit some things to cross namespace boundaries

```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: cross-namespace
  namespace: faces
spec:
  rules:
    - matches:
        - path: /magic/path
      backendRefs:
        - name: magic
          namespace: magic-namespace
```







ReferenceGrant 👷

```yaml
apiVersion: gateway.networking.k8s.io/v1beta1
kind: ReferenceGrant
metadata:
  name: magic-grant
  namespace: magic-namespace
spec:
  from:
  - group: gateway.networking.k8s.io
    kind: HTTPRoute
    namespace: faces
  to:
  - group: ""
    kind: Service
```

```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: cross-namespace
  namespace: faces
spec:
  rules:
    - matches:
        - path: /magic/path
      backendRefs:
        - name: magic
          namespace: magic-namespace
```





Policy attachment 🤷🪱

Policy attachment allows modifying the behavior of resources **without** modifying the resources.

- Very powerful, but **very** complex
- Use only with extreme caution!
- In particular, this isn't great for things Ana needs to do often

```yaml
apiVersion: v1
kind: Service
metadata:
  name: weird-service
  namespace: weird-namespace
spec: ...
```

```yaml
apiVersion: gateway.networking.k8s.io/v1beta1
kind: MoonPhasePolicy
metadata:
  name: full-moon-only
  namespace: weird-namespace
spec:
  targetRef:
    - group: ""
      kind: Service
      name: weird-service
  allowedMoonPhases:
    - full
```

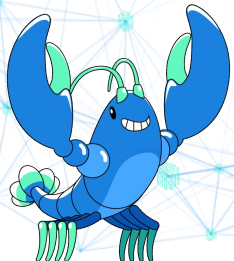




BUOYANT
SERVICE MESH
ACADEMY

gateway api

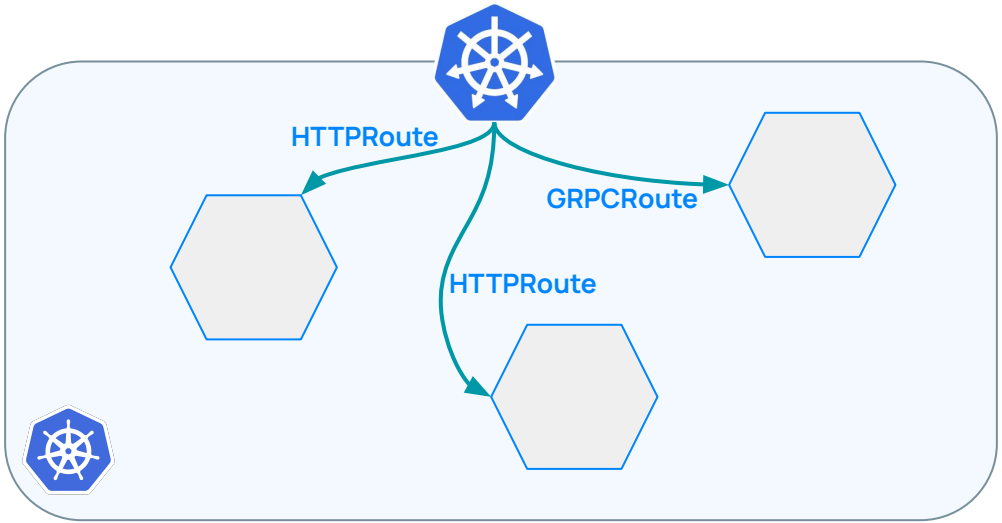
Great, what about the service mesh?





Gateway API for mesh

- As it happens, these concepts work pretty much the same wherever routing happens!

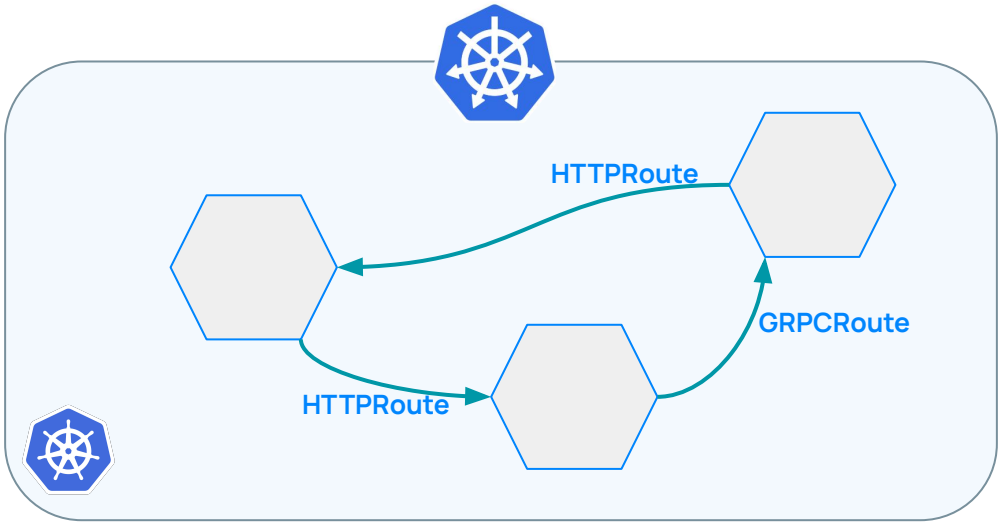






Gateway API for mesh

- As it happens, these concepts work pretty much the same wherever routing happens!
- The difference isn't in *what a request is*, it's in *where it's presented*







Mesh routing 👩‍💻

Requests in the mesh arrive at a Service, rather than a Gateway

- This is reflected in the `parentRefs`; the rest is the same
- Of course, the mesh might not support everything the gateway does
- For example, meshes will probably not support CORS

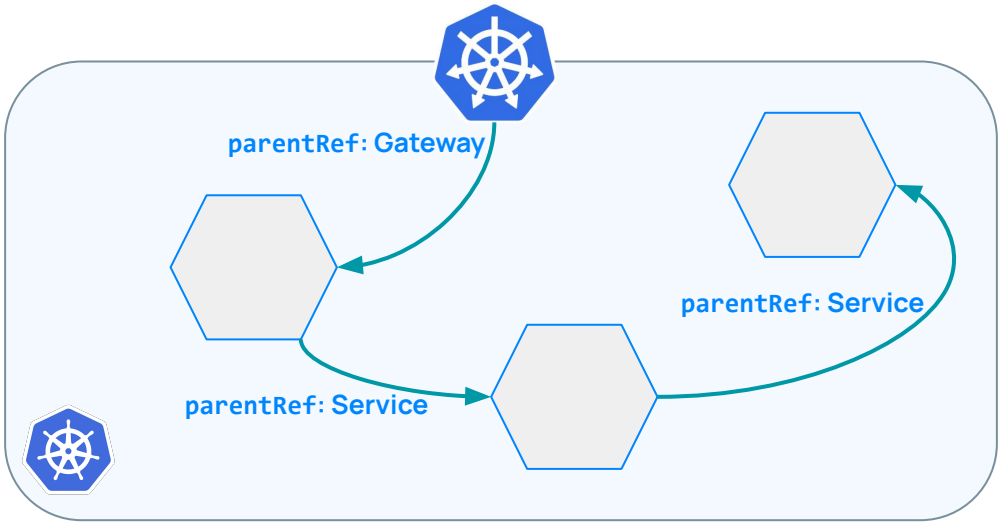
```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: simplest-smiley-route
  namespace: faces
spec:
  parentRefs:
    - kind: Service
      group: ""
      name: smiley
      port: 80
  hostnames:
    - smiley.example.com
  rules:
    - backendRefs:
      - name: smiley
        port: 80
```





Gateway API for mesh

- The end result is that the `parentRefs` define *classes of traffic*, so choose your parents wisely!
- (Also, we'll have to define new kinds of parents for new classes of traffic…)

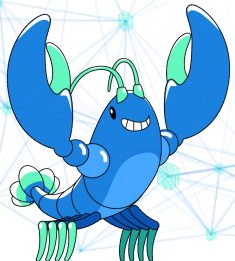




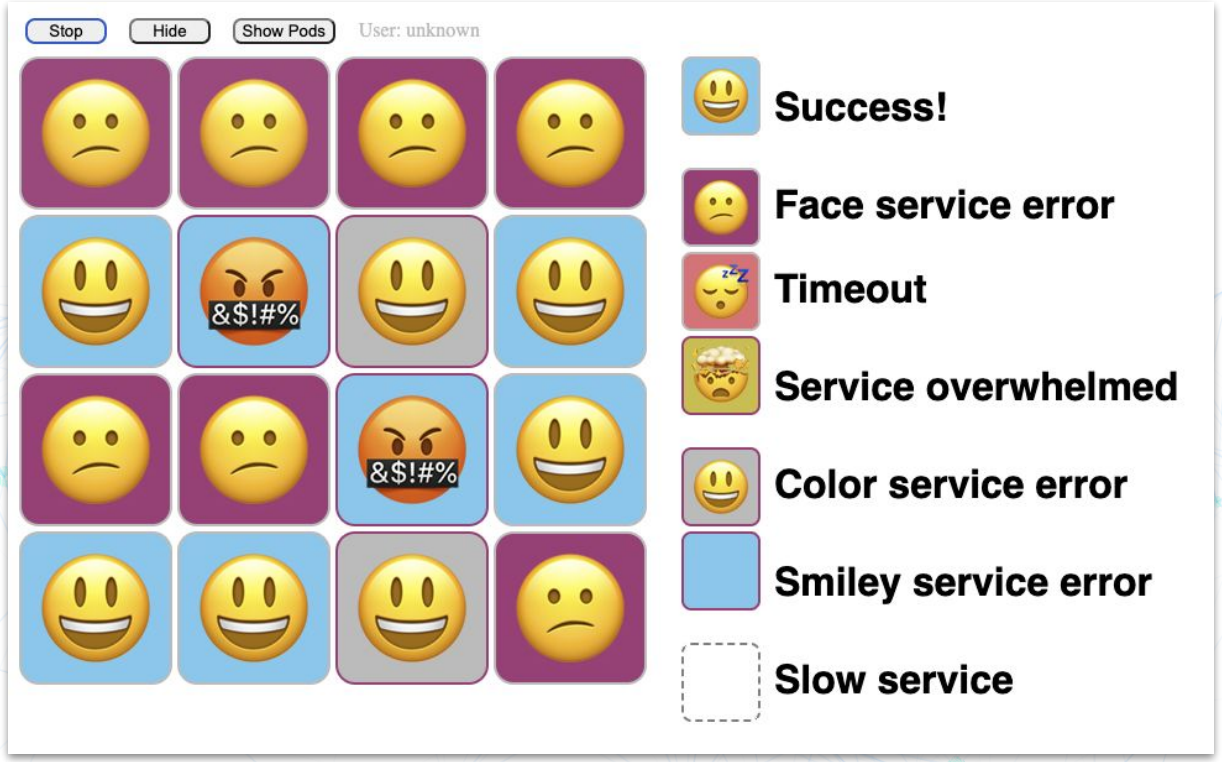
BUOYANT
SERVICE MESH
ACADEMY

gateway api

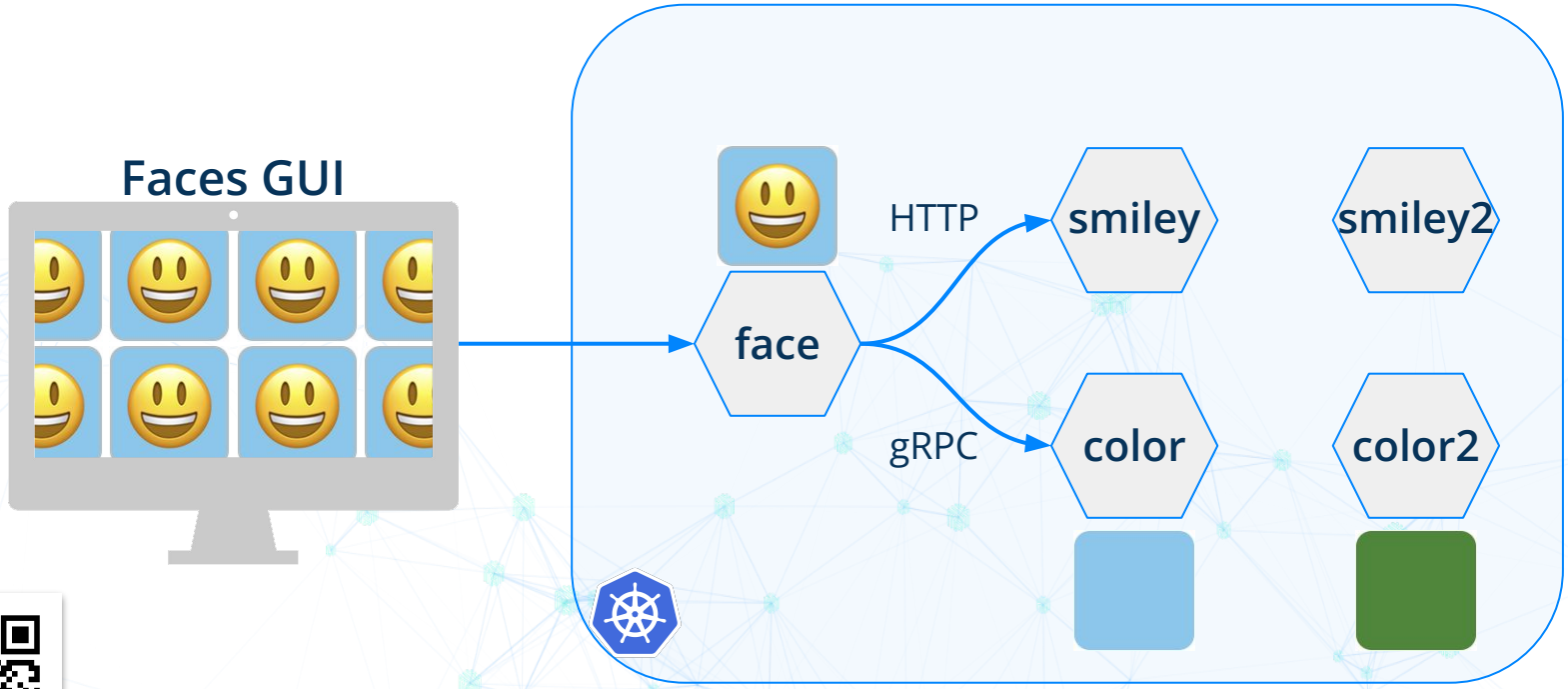
Demo Architecture



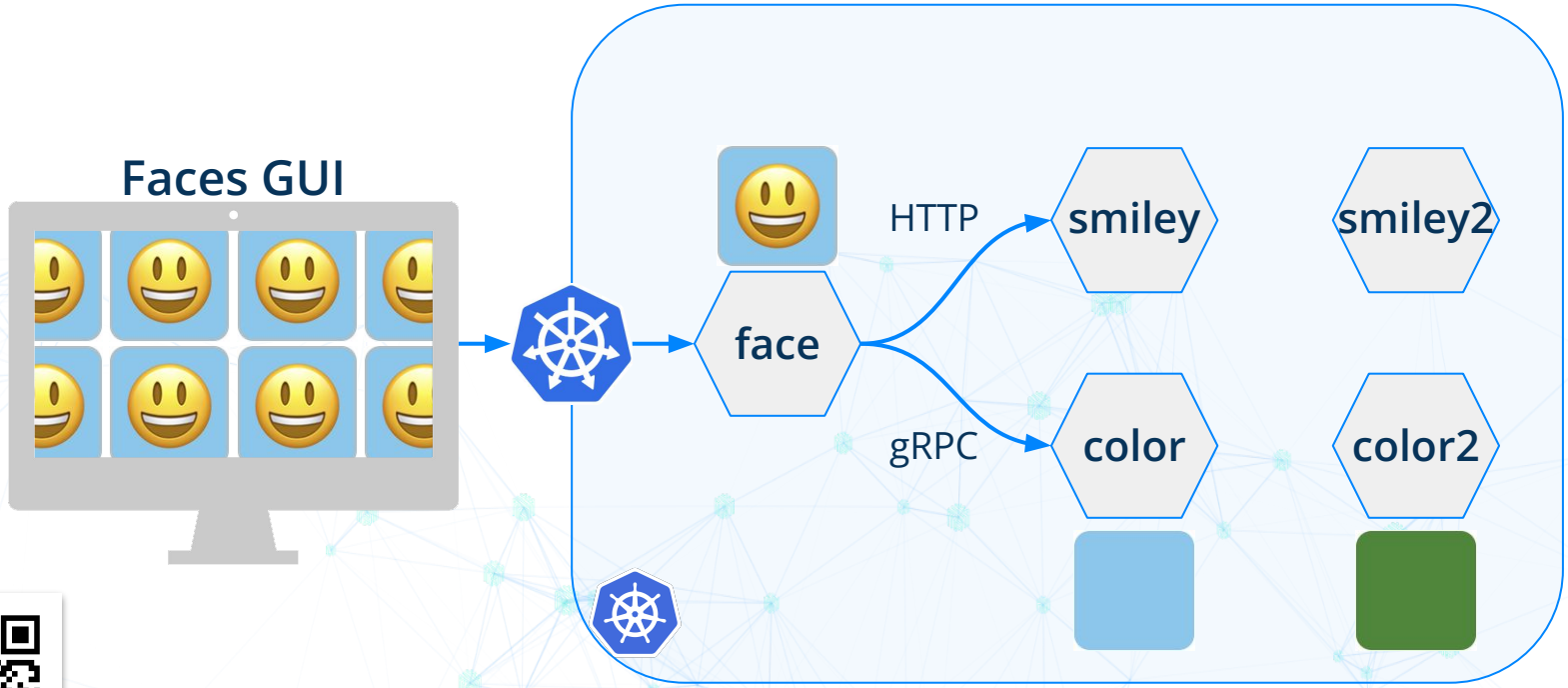
Faces (http://github.com/BuoyantIO/faces-demo)



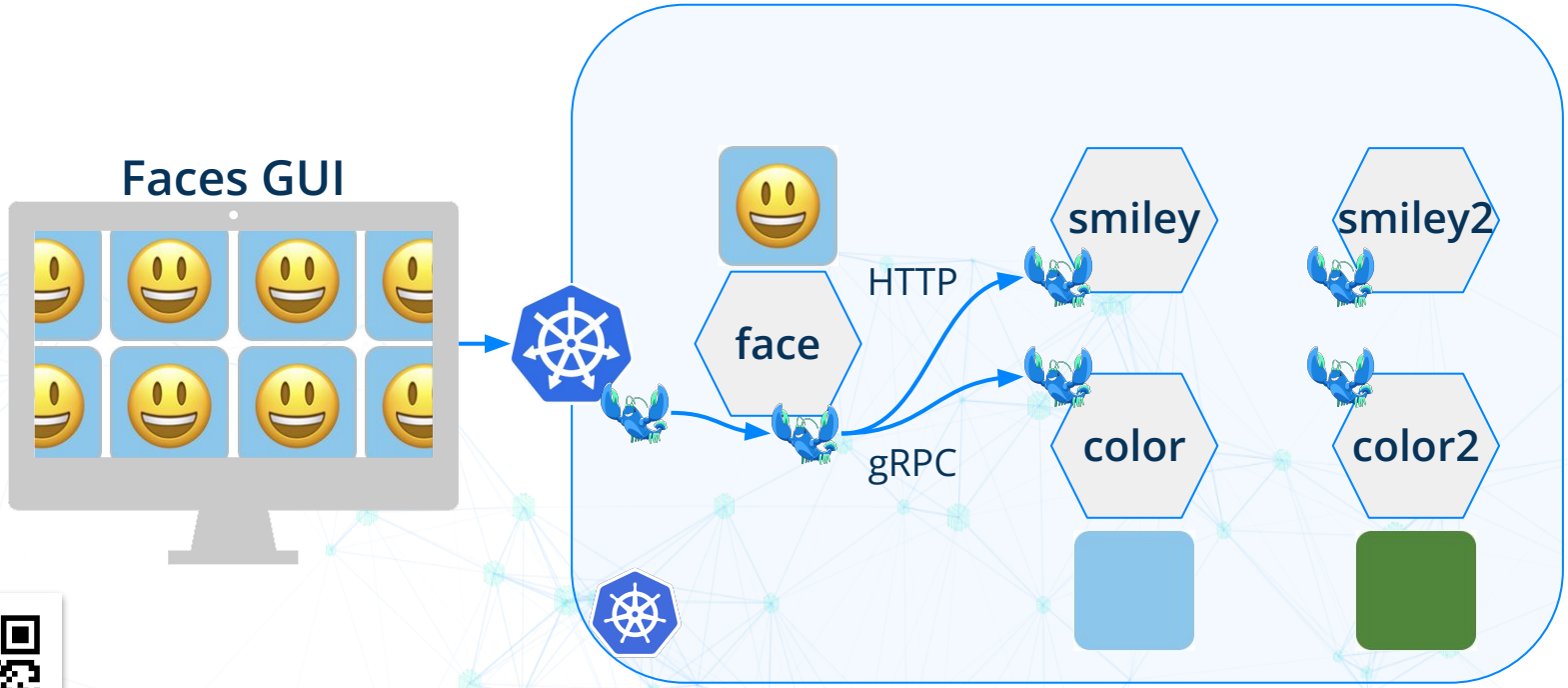
Faces (http://github.com/BuoyantIO/faces-demo)



Faces (http://github.com/BuoyantIO/faces-demo)



Faces (http://github.com/BuoyantIO/faces-demo)

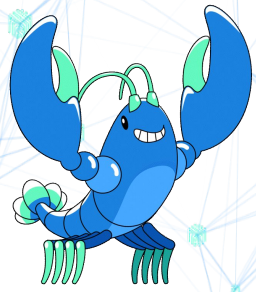




BUOYANT
SERVICE MESH
ACADEMY

gateway api

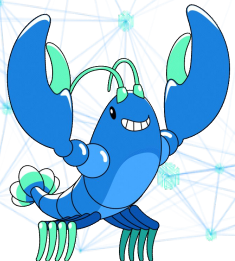
DEMO 🤞



BUOYANT
SERVICE MESH
ACADEMY

gateway api

What's going well?





Many things are going well!

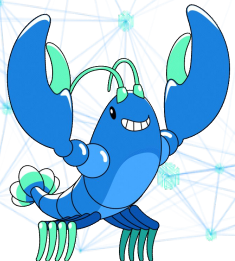
- Gateway API is the “most collaborative” API in Kubernetes
  - Over 250 contributors
- It’s the first CRD API to be considered “part of Kubernetes”
- It’s (mostly) well-specified and unambiguous
- It’s already much more feature-rich than Ingress was
- Active community shipping faster than core Kubernetes



BUOYANT
SERVICE MESH
ACADEMY

gateway api

What's going less well?





Some things are going less well

- Ana's life with Gateway API is tougher than Chihiro's and Ian's
  - We're working on this!
- Understanding use cases well enough to standardize them is *very hard* (especially at L4)
- Extensions are much less straightforward than we'd like
  - We haven't really avoided annotation hell, we've just created policy hell
- We've struggled with "Experimental" APIs





Some things are going less well

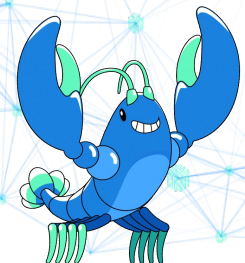
- Lifecycle management of CRD APIs is currently very problematic
  - At this point, if you're building on top of Gateway API, **please avoid installing the Gateway API CRDs** in an attempt to help your users!
    - Gateway API in practice is very much like a "core API" today
    - Ideally, your Kubernetes platform (e.g. OpenShift, etc) should be providing these for you.
      - There is still upstream work to be done to make this better and more consistent across platforms
    - We have found over the past year or two, especially, that doing so makes life *much harder* for interoperating projects to use Gateway API
  - Beware of "Unknown" or "Dead Fields" when using CRDs provided
    - We're working on this problem in upstream
    - for today make sure you are clear on exactly what version of the API is present



BUOYANT
SERVICE MESH
ACADEMY

gateway api

Should you use this?





Should you use Gateway API?

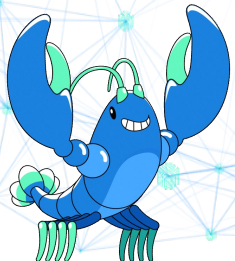
- If you're happy with Ingress and annotations, or the Service API you can keep using those they are not going away.
- BUT:
  - More and more projects are requiring Gateway API (for example, Linkerd, OpenShift*, and Istio Ambient all *require* Gateway API)
  - If you need features beyond basic Ingress, you should go ahead and make the jump
  - For new projects, likely you'll want to *start with it*



BUOYANT
SERVICE MESH
ACADEMY

gateway api

How can you get involved?





Getting involved

- #sig-network-gateway-api on slack.k8s.io
- https://github.com/kubernetes-sigs/gateway-api
- https://gateway-api.sigs.k8s.io/
- and of course,
  https://github.com/BuoyantIO/service-mesh-academy/tree/main/gateway-api-101



SERVICE MESH ACADEMY

Tell us how we can improve!

Your feedback matters!

(We promise it won't take more than a few minutes, and it will help us tremendously — thank you! 🙂)



BUOYANT
Creators of LINKERD

Buoyant Enterprise for LINKERD

Rust-based network security and reliability for modern applications. Built on open source and designed for the enterprise.

- ➔ Zero-trust security and compliance across your entire network
- ➔ Global traffic management and control
- ➔ Full L7 application observability
- ➔ Built for the enterprise

Learn more & try it for free at buoyant.io/**enterprise-linkerd**

BUOYANT
Creators of LINKERD



Updated Course!

BUOYANT
Creators of LINKERD

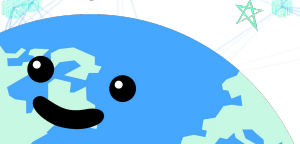


Get Certified!
Service Mesh 101, Online Course

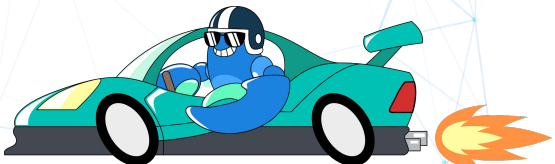
With hands-on LINKERD labs at learn.buoyant.io









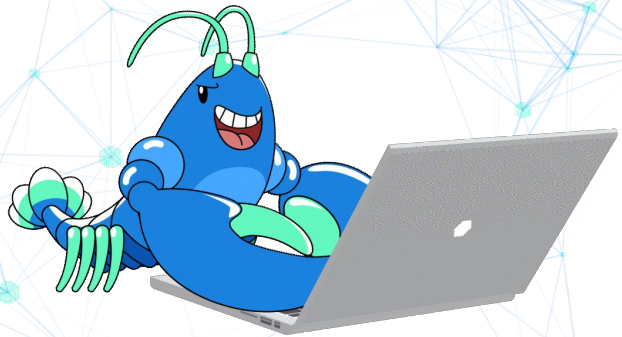


SERVICE MESH ACADEMY

Up Next on MAY 15 (in April we're at KubeCon, come say hi!)

Linkerd 2.18

SIGN UP TODAY!
buoyant.io/sma



BUOYANT
SERVICE MESH
ACADEMY

gateway api

Q&A

