KubeCon | CloudNativeCon

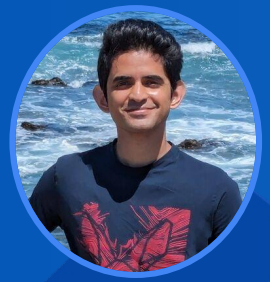
North America 2024

With Great Flexibility Comes Great Complexity:

Inspect Your Gateway API Configuration

Gaurav Ghildiyal @ Google, Mattia Lavacca @ Kong

KubeCon CloudNativeCon
North America 2024



Gaurav Ghildiyal @ Google
gwctl maintainer
gauravkghildiyal

Mattia Lavacca @ Kong
Gateway API maintainer
mlavacca

Gateway API Mastery: Our Journey Today

- Aim: Simplify Your Gateway API Experience
- Setting the stage:
  - Ingress: The Precursor to Gateway API
  - Limitations of Ingress
  - Gateway API: Solutions and New Challenges
- Deep dive:
  - Exploring Gateway API complexities
  - Effective strategies to overcome these challenges

A Quick Primer on Ingress

Ingress: what's good

- Kubernetes core API
- Simplicity

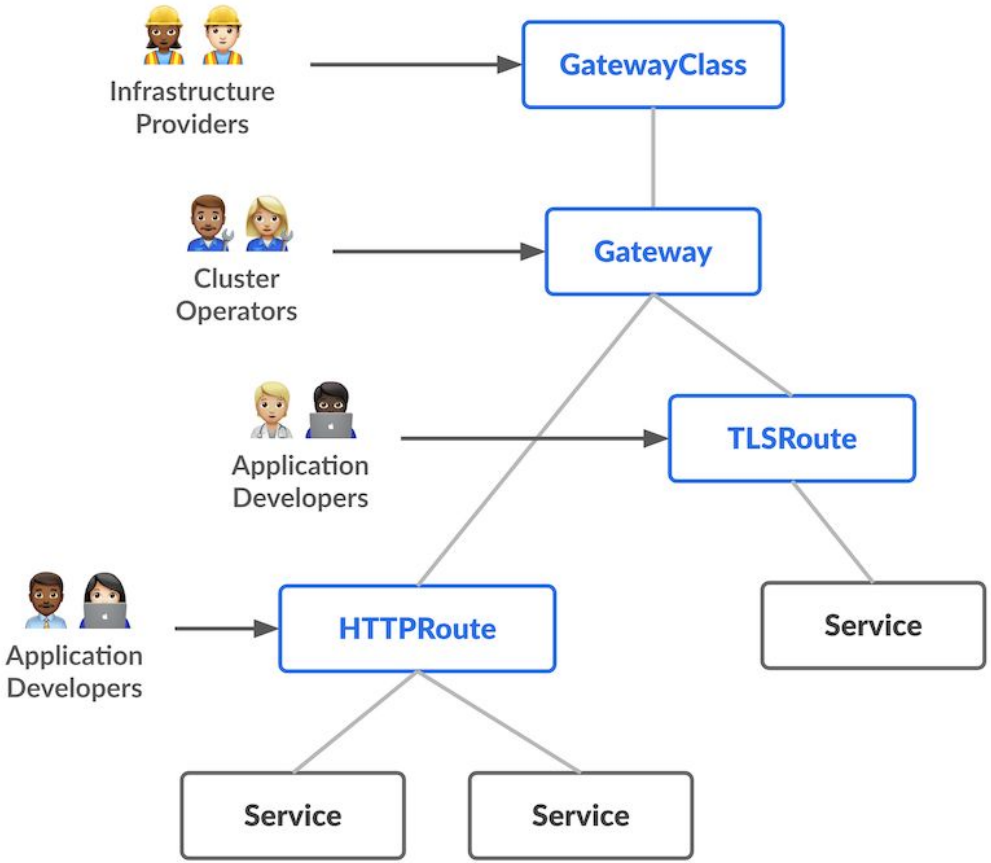
```yaml
kind: Ingress
metadata:
  name: login-ing
spec:
  ingressClassName: <your-implementation>
  rules:
  - host: foo.example.com
    http:
      paths:
      - path: /login
        pathType: Prefix
        backend:
          service:
            name: auth-svc
            port:
              number: 8080
```

Ingress: what's bad

- Lack of core features
- Custom extensions everywhere
- Extensions are not portable
- Lack of protocol diversity
- Insufficient permissions model

Gateway API: what's good

- Persona focused model
- Flexible and Extensible
- Portable
- Large community support



Gateway API: what's good (cont.)

HTTPRoute

```yaml
kind: HTTPRoute
metadata:
  name: login
spec:
  parentsRef:
  - name: gke-external
  hostnames:
  - foo.example.com
  rules:
  - matches:
    - path:
        type: Prefix
        value: /login
    backendRefs:
    ...
```

Ingress

```yaml
kind: Ingress
metadata:
  name: login-ing
spec:
  ingressClassName: gke-external
  rules:
  - host: foo.example.com
    http:
      paths:
      - path: /login
        pathType: Prefix
        backend:
          service:
            name: auth-svc
            port:
              number: 8080
```

Gateway API: what's good (cont.)

HTTPRoute

```yaml
kind: HTTPRoute
metadata:
  name: login
spec:
  parentsRef:
  - name: gke-external
  hostnames:
  - foo.example.com
  - bar.example.org
  rules:
  - matches:
    - path:
        type: Prefix
        value: /login
    backendRefs:
    ...
```

Ingress

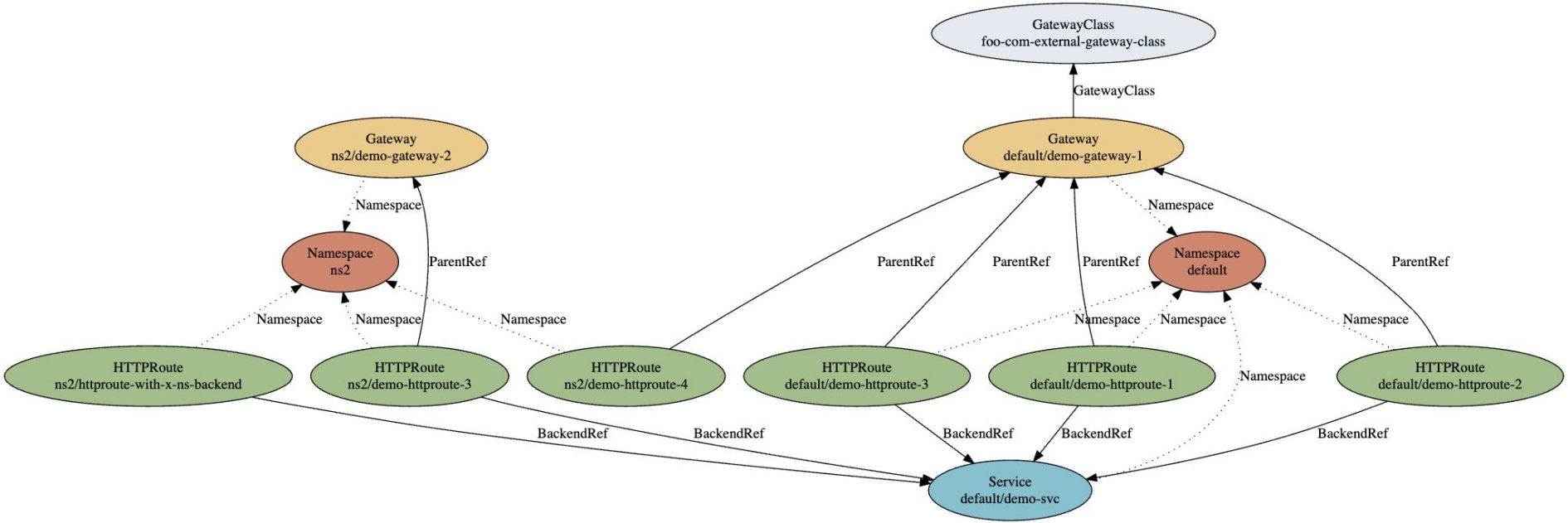
```yaml
kind: Ingress
metadata:
  name: login-ing
spec:
  ingressClassName: gke-external
  rules:
  - host: foo.example.com
    http:
      paths:
      - path: /login
        pathType: Prefix
        backend:
          service:
            name: auth-svc
            port:
              number: 8080
```

This is the way

Well, not that fast...

Gateway API complexity

Gateway API: what's bad



Gateway API entities

- GatewayClass
- Gateway
- Routes
  - HTTPRoutes
  - GRPCRoutes
- ReferenceGrant
- Policies

Is it overwhelming?

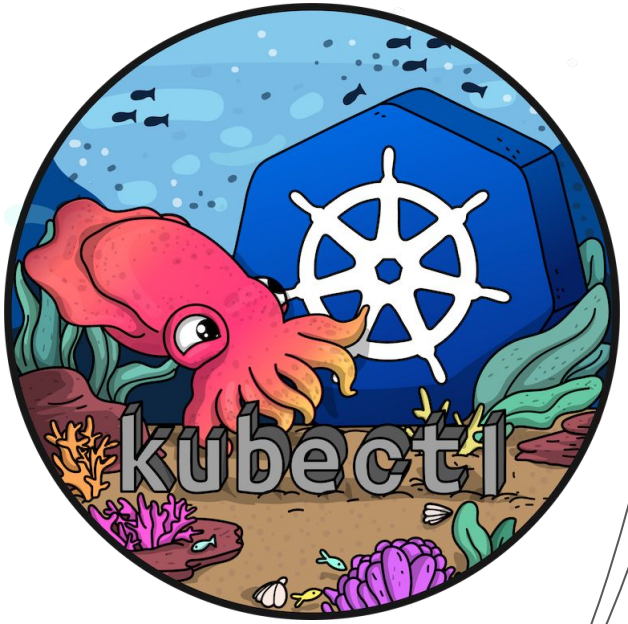
Introducing **gwctl**

gwctl

- What is **gwctl**?
  - A command-line tool designed specifically for Gateway API.
- Pronounced: *gateway-cuttle*
- Initially bundled with the Gateway API repo, now moved to kubernetes-sigs/gwctl
- First release: v0.1.0

sigs.k8s.io/gwctl

Why kubectl isn't enough for Gateway API?



Kubernetes Core APIs

- Pods
- Deployments
- Services
- ConfigMaps
- CRDs

gwctl understands…

Gateway API
(Official Kubernetes APIs)

- GatewayClass
- HTTPRoute
- Gateway
- ReferenceGrant

What can you do with gwctl?

What can you do with gwctl?

- `gwctl get gatewayclasses`
- `gwctl get httproutes -n some-namespace`
- `gwctl get gateways -l version=v1`
- `gwctl get svc -o yaml`
- `gwctl get gateways/demo-gateway-1 httproutes/demo-httproute-1`
- `gwctl get svc -l version=v1 -n some-namespace -o json`
- `gwctl apply -f /path/to/config.yaml`
- `cat ... | gwctl apply -f -`
- `gwctl delete -f /path/to/config.yaml`
- `gwctl delete gateways my-gateway`

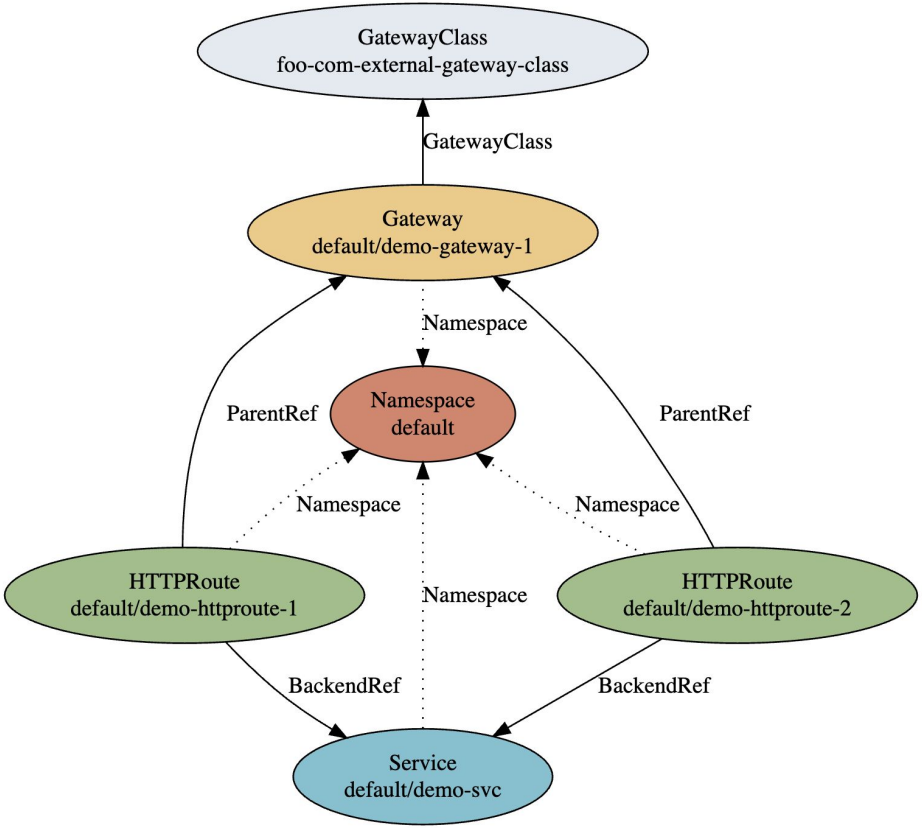
Just another kubectl?

The similarities are only the beginning...

Unmasking Hidden Relationships: Who's Referencing What?

Unmasking Hidden Relationships: Who's Referencing What?

- Difficulty in identifying which resources reference a given resource.
- kubectl lacks the ability to show these connections.



gwctl to the rescue: Revealing Reverse References

- gwctl makes it easy to see reverse references.
- -o wide provides a summary.

```
$ gwctl get service -o wide
NAMESPACE  NAME        TYPE     AGE   REFERRED BY ROUTES                                  POLICIES
default    demo-svc    Service  12m   default/demo-httproute-1, default/demo-httproute-2  0
default    kubernetes  Service  113m  None                                                0

$ gwctl get gateways -o wide
NAMESPACE  NAME           CLASS                          ADDRESSES  PORTS  PROGRAMMED  AGE  POLICIES  HTTPROUTES
default    demo-gateway-1  foo-com-external-gateway-class            80     Unknown     18m  0         2

$ gwctl get gatewayclasses -o wide
NAME                          CONTROLLER                     ACCEPTED  AGE  GATEWAYS
foo-com-external-gateway-class  foo.com/external-gateway-class  Unknown   18m  1
```

gwctl to the rescue: Revealing Reverse References

- gwctl makes it easy to see reverse references.
- -o wide provides a summary.

```
$ gwctl get service -o wide
NAMESPACE  NAME        TYPE     AGE    REFERRED BY ROUTES                                          POLICIES
default    demo-svc    Service  6h13m  default/demo-httproute-1, default/demo-httproute-2 + 2 more  1
default    kubernetes  Service  3d9h   None                                                        0
```

```
$ gwctl get gateways -o wide
NAMESPACE  NAME            CLASS                          ADDRESSES  PORTS  PROGRAMMED  AGE    POLICIES  HTTPROUTES
default    demo-gateway-1  foo-com-external-gateway-class             80     Unknown     6h13m  2         4
```

```
$ gwctl get gatewayclasses -o wide
NAME                          CONTROLLER                      ACCEPTED  AGE    GATEWAYS
bar-com-internal-gateway-class  bar.baz/internal-gateway-class  Unknown   6h14m  1
```

gwctl to the rescue: Revealing Reverse References

- gwctl describe reveals detailed referencing information.
- Example:
  - **`gwctl describe gateways`**

```
$ gwctl describe gateways
Name: demo-gateway-1
Namespace: default
...
AttachedRoutes:
  Kind        Name
  ----        ----
  HTTPRoute   default/demo-httproute-1
  HTTPRoute   default/demo-httproute-2
Backends:
  Kind      Name
  ----      ----
  Service   default/demo-svc
  Service   default/demo-svc
```

gwctl to the rescue: Revealing Reverse References

- gwctl describe reveals detailed referencing information.
- Example:
  - `gwctl describe gateways`
  - **`gwctl describe svc`**

```
$ gwctl describe svc
Name: demo-svc
...
ReferencedByRoutes:
  Kind        Name
  ----        ----
  HTTPRoute   default/demo-httproute-1
  HTTPRoute   default/demo-httproute-2
...
```

The Perils of Broken Links in Gateway API

The Perils of Broken Links in Gateway API

- Gateway API relies on accurate references between resources.
- A missing Gateway, Service, or other resource is invalid.

```yaml
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
...
 rules:
 - matches:
   - path:
       type: PathPrefix
       value: /example
   backendRefs:
   - name: example-svc
     port: 80
```

Service does not exist

gwctl describe: Exposing Broken Links

- gwctl describe automatically detects invalid references.

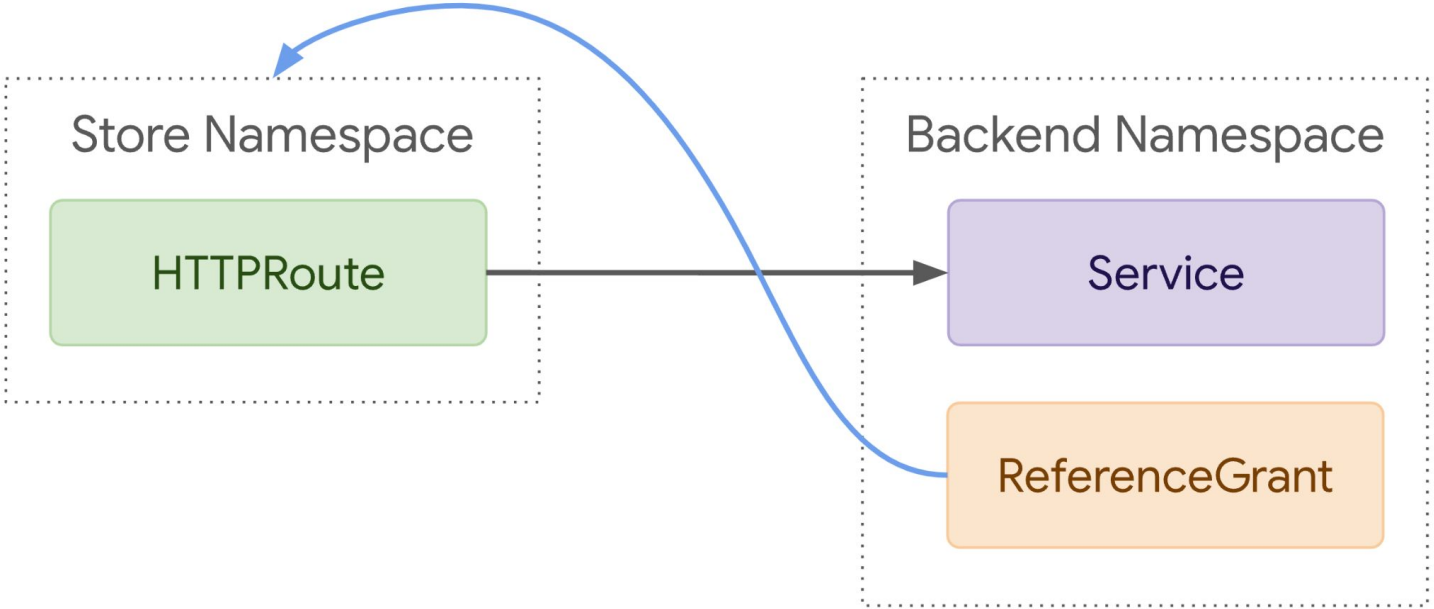
```
$ gwctl describe gateways
Name: demo-gateway-1
Namespace: default
...
Analysis:
- Gateway(.gateway.networking.k8s.io) "default/demo-
gateway-1" references a non-existent
  GatewayClass(.gateway.networking.k8s.io) "random-gateway-
class"
...

$ gwctl describe httproutes
Name: demo-httproute-1
Namespace: default
...
Analysis:
- HTTPRoute(.gateway.networking.k8s.io) "default/demo-
httproute-1" references a non-existent
  Service "default/demo-svc-100"
...
```

Managing Cross-Namespace Access

Managing Cross-Namespace Access

- ReferenceGrants control access to resources across namespaces.



gwctl describe: Simplifying ReferenceGrant Validation

- gwctl describe analyzes ReferenceGrant permissions.

```
$ gwctl describe httproute -n store-ns
Name: demo-httproute-3
Namespace: store-ns
...
Analysis:
- HTTPRoute(.gateway.networking.k8s.io) "store-ns/demo-
httproute-3" is not permitted to
  reference Service "backend-ns/demo-svc"
...
```

Policies: The Hidden Power of Gateway API

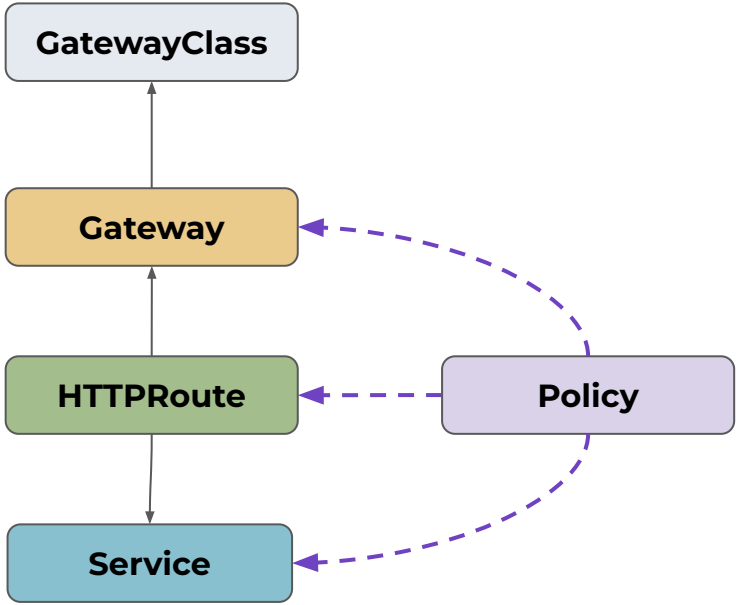
Policies: The Hidden Power of Gateway API

- Policies provide a powerful mechanism for customization and extending functionality. (GEP-713)

- But managing policies can be challenging without the right tools.

Policies in the Shadows: The Challenge of Discovery

- Policies can be attached to various resources, making them hard to track.
- Understanding which policies are applied to a resource is crucial for troubleshooting and management.



gwctl: Policies as a native resource

- `gwctl get policies`

```
$ gwctl get policies
NAME                          KIND                                          TARGET NAME       TARGET KIND  POLICY TYPE  AGE
tls-upstream-dev              BackendTLSPolicy.gateway.networking.k8s.io                                     Direct       15m
demo-health-check-1           HealthCheckPolicy.foo.com                     demo-gateway-1    Gateway      Direct       15m
demo-retry-policy-1           RetryOnPolicy.foo.com                         demo-gateway-1    Gateway      Direct       15m
demo-retry-policy-2           RetryOnPolicy.foo.com                         demo-httproute-2  HTTPRoute    Direct       15m
demo-tls-min-version-policy-1 TLSMinimumVersionPolicy.baz.com               demo-httproute-1  HTTPRoute    Direct       15m
demo-tls-min-version-policy-3 TLSMinimumVersionPolicy.baz.com               demo-svc          Service      Direct       15m
```

gwctl: Policies as a native resource

- `gwctl get policycrds`
- Gateway API Implementor: Label CRD with `gateway.networking.k8s.io/policy` for gwctl recognition

```
$ gwctl get policycrds
NAME                                                POLICY TYPE  SCOPE       AGE
backendlbpolicies.gateway.networking.k8s.io         Direct       Namespaced  3d3h
backendtlspolicies.gateway.networking.k8s.io        Direct       Namespaced  3d3h
healthcheckpolicies.foo.com                         Direct       Namespaced  3d3h
retryonpolicies.foo.com                             Direct       Namespaced  3d3h
timeoutpolicies.bar.com                             Inherited    Cluster     3d3h
```

gwctl describe: Revealing Attached Policies

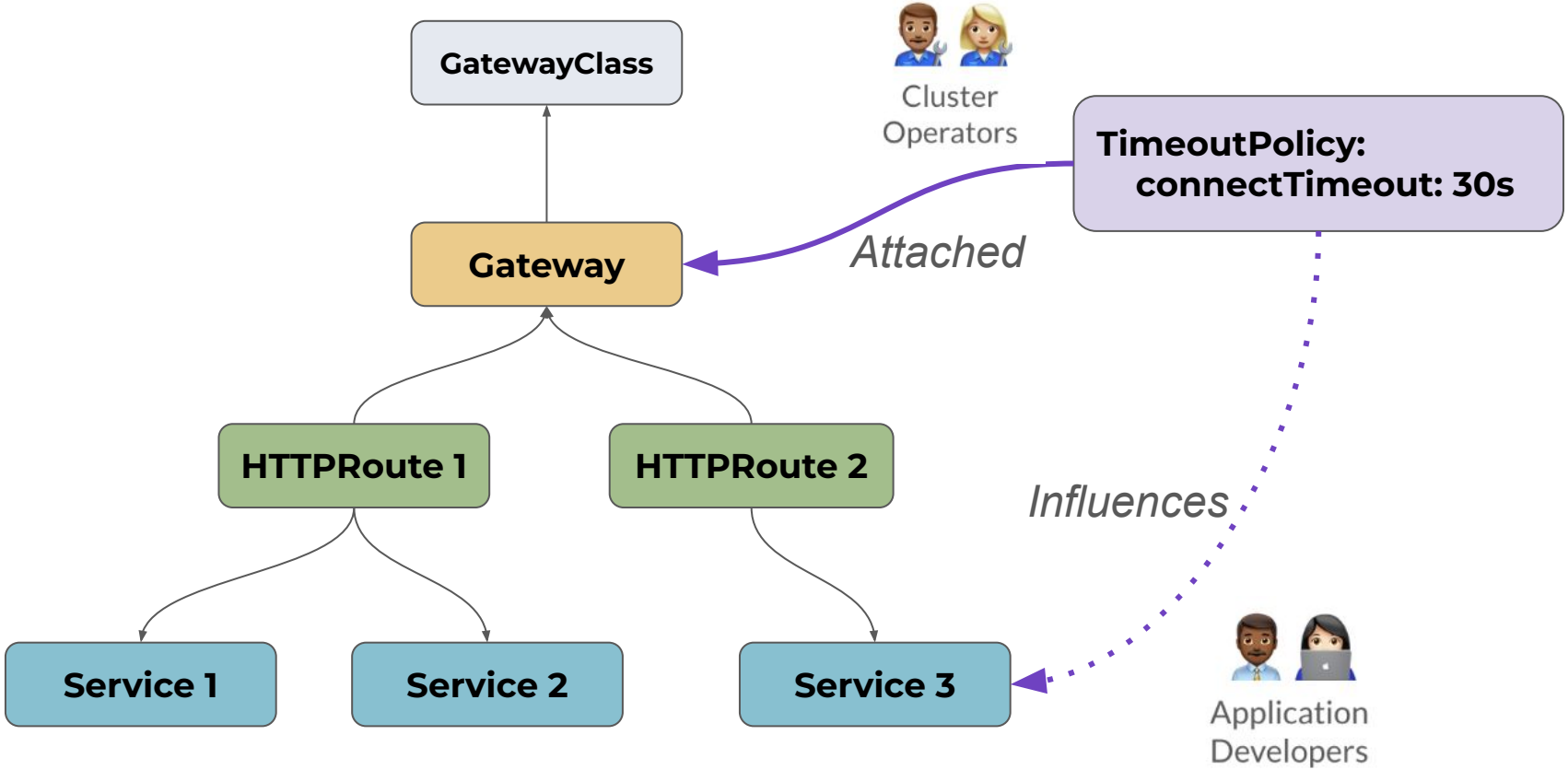
- gwctl describe clearly shows the policies directly attached to a resource.
- Gain insights into how policies are influencing resource behavior.

```
$ gwctl describe httproute
Name: demo-httproute-1
Namespace: default
...
DirectlyAttachedPolicies:
  Type                              Name
  ----                              ----
  TLSMinimumVersionPolicy.baz.com   default/demo-tls-min-version-policy-1
...
```

Policy Inheritance

- Policies can be inherited from parent resources like GatewayClasses and Gateways.
- Tracing the origin of inherited policies can be tedious.

Policy Inheritance

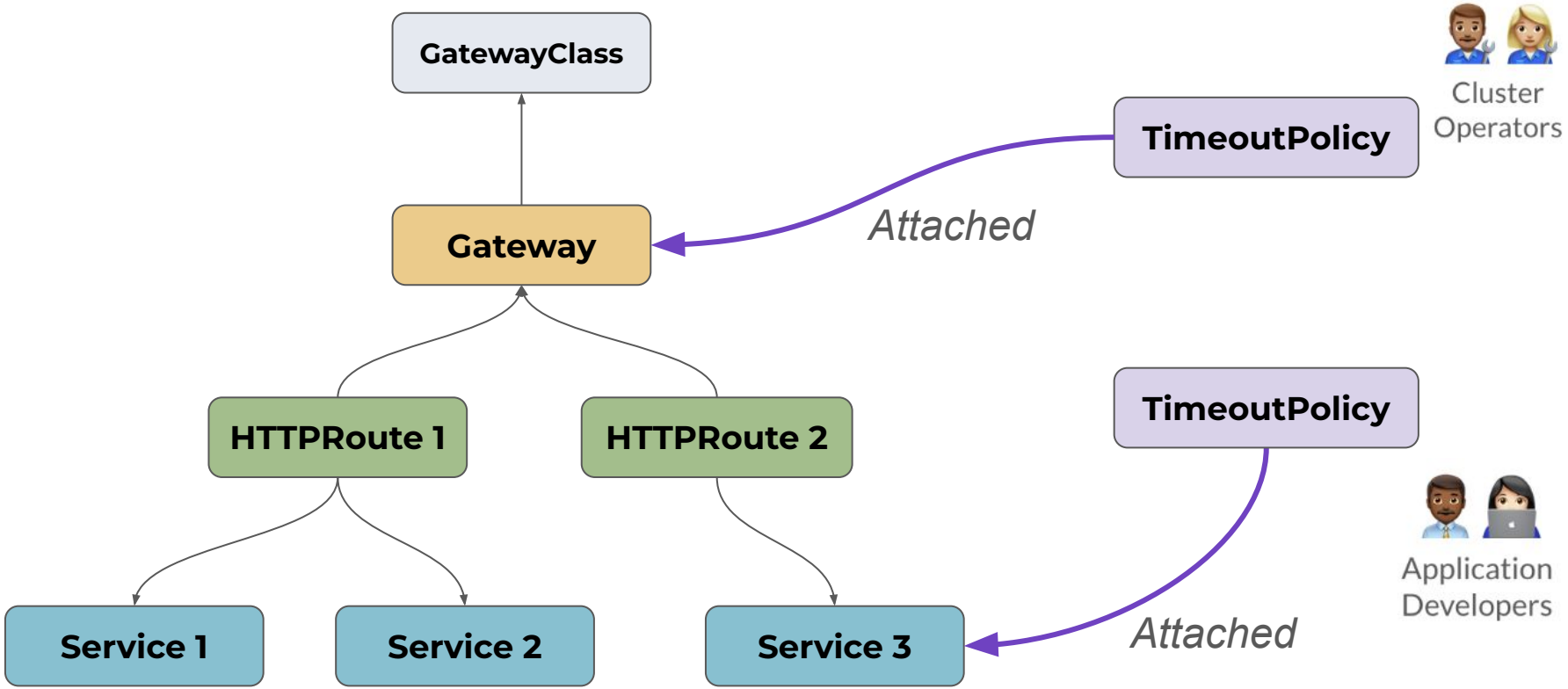


gwctl describe: Untangling Policy Inheritance

- gwctl describe helps trace the origin of inherited policies.
- Understand the complete policy chain affecting a resource.

```
$ gwctl describe svc
Name: demo-svc
Namespace: default
...
DirectlyAttachedPolicies:
  Type                              Name
  ----                              ----
  TLSMinimumVersionPolicy.baz.com   default/demo-tls-min-version-policy-3
InheritedPolicies:
  Type                   Name                                  Target Kind   Target Name
  ----                   ----                                  -----------   -----------
  TimeoutPolicy.bar.com  demo-timeout-policy-on-gatewayclass   GatewayClass  foo-com-external-gateway-class
  TimeoutPolicy.bar.com  demo-timeout-policy-on-namespace      Namespace     default
...
```

Policy Inheritance: Multiple Policies



Effective Policies: A Puzzle of Defaults and Overrides

- Effective policies are the final set of rules applied after considering defaults, and overrides.
- Manually calculating the effective policy can be a complex and error-prone process.

gwctl describe: Solving the Effective Policy Puzzle

- gwctl describe calculates and displays the effective policies.

```
$ gwctl describe svc
Name: demo-svc
Namespace: default
...
InheritedPolicies:
  Type                  Name                          Target Kind   Target Name
  ----                  ----                          -----------   -----------
  TimeoutPolicy.bar.com  demo-timeout-policy-on-gateway  Gateway       demo-gateway-1
EffectivePolicies:
  Gateway.gateway.networking.k8s.io/default/demo-gateway-1:
    TimeoutPolicy.bar.com:
      connectTimeout: 30s
  Gateway.gateway.networking.k8s.io/default/demo-gateway-2:
    TimeoutPolicy.bar.com:
      connectTimeout: 45s
    RetryPolicy.foo.com:
      maxRetries: 5
...
```

per-gateway

per-policytype

Check before you deploy

Check before you deploy

- Feeling a bit uneasy?
- Need a pre-flight check?

Do a dry-run with gwctl analyze

gwctl analyze performs dry-run analysis of YAML files.

Output contains:

- Summary
- Potential Issues Introduced
- Existing Issues Fixed
- Existing Issues Unchanged

```
$ gwctl analyze -f /tmp/gwctl-test.yaml

Analyzing /tmp/gwctl-test.yaml...

Summary:

        - Created referencegrants/my-reference-grant in namespace default
        - Updated services/demo-svc in namespace default

Potential Issues Introduced
(These issues will arise after applying the changes in the analyzed file.):

        None.

Existing Issues Fixed
(These issues were present before the changes but will be resolved after applying them.):

        - HTTPRoute.gateway.networking.k8s.io/ns2/demo-httproute-3:
          HTTPRoute(.gateway.networking.k8s.io) "ns2/demo-httproute-3" is not
          permitted to reference Service "default/demo-svc":

Existing Issues Unchanged
(These issues were present before the changes and will remain even after applying them.):

        - HTTPRoute.gateway.networking.k8s.io/ns2/demo-httproute-4:
          HTTPRoute(.gateway.networking.k8s.io) "ns2/demo-httproute-4"
          references a non-existent Service "ns2/demo-svc":
...
```

Drowning in a Sea of YAML?

Drowning in a Sea of YAML?

- Reading through a bunch of YAML to figure things out is difficult
- What a big-picture view?

gwctl graph: Visualizing Gateway API

- Generate graph representations of your Gateway API configurations.
- Use DOT language for visualization with tools like Graphviz.
- Example: `gwctl get gateway -o graph`

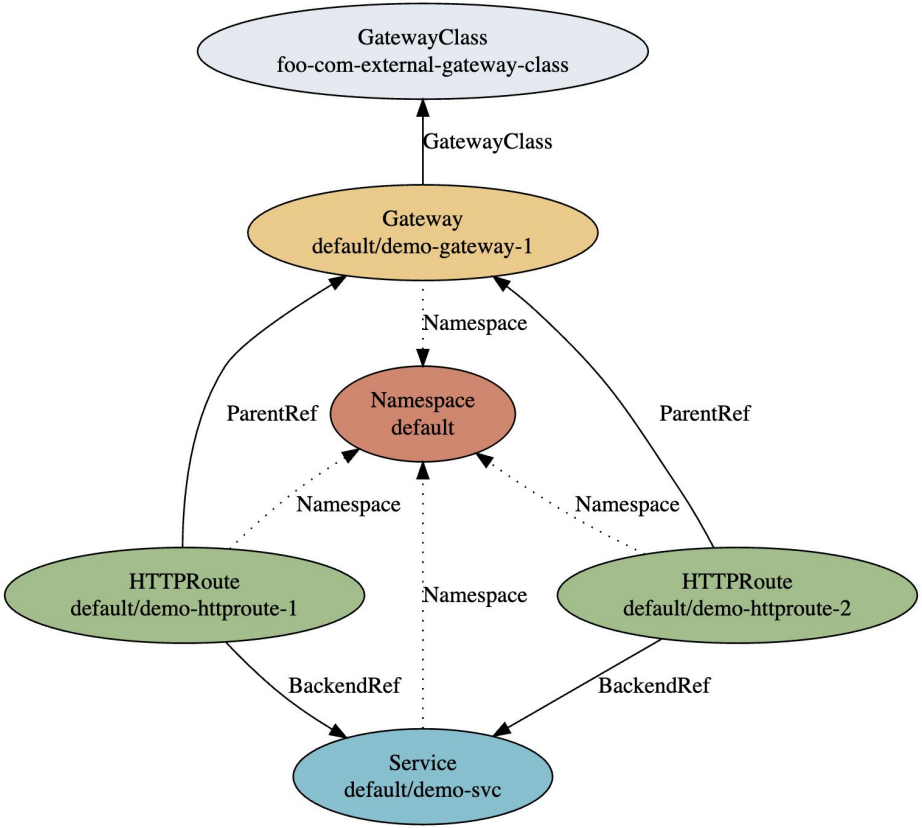
```
$ gwctl get gateway -o graph
digraph "" {
  # ... (Nodes and edges representing the relationships)
}
```

gwctl graph: Gaining a Bird's-Eye View

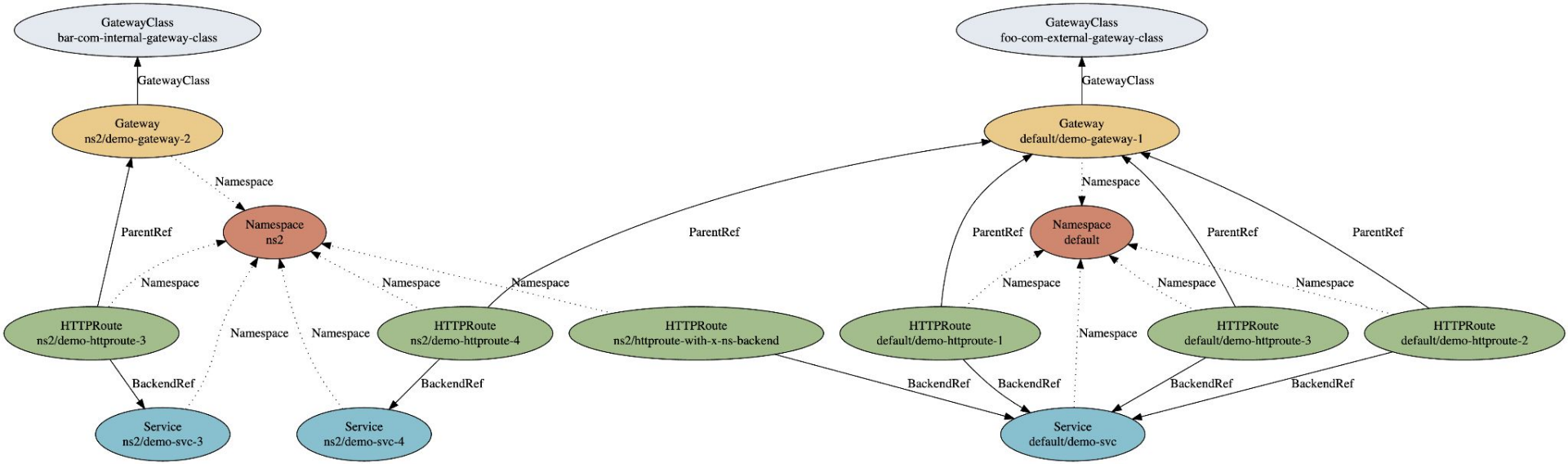
- Generate graph representations of your Gateway API configurations.
- Use DOT language for visualization with tools like Graphviz.
- Example: `gwctl get gateway -o graph`

```
$ gwctl get gateway -o graph
digraph "" {
 # ... (Nodes and edges representing the relationships)
}
```

gwctl graph: Sample 1



gwctl graph: Sample 2



Recap

- Limitations of Ingress API and the evolution to Gateway API.
- Complexities of Gateway API multi-resource model.
- Need for gwctl
- Solutions to common challenges:
  - Where is your resource getting used?
    - **-o wide** or **gwctl describe**

gwctl empowers you today, while we build a more integrated tomorrow.

    - **gwctl describe** -> **DirectlyAttachedPolicies, InheritedPolicies, EffectivePolicies**
  - How can you sanity check your changes?
    - **gwctl analyze**
  - How to visualize your Gateway API resources?
    - **-o graph**

Powered by the Community

Thank You, gwctl Contributors!

- arukiidou (@arukiidou)
- Candace Holman (@candita)
- Christine Kim (@xtineskim)
- Dave Protasowski (@dprotaso)
- Dennis Zhou (@deszhou)
- Devaansh Kumar (@Devaansh-Kumar)
- Gaurav Ghildiyal (@gauravkghildiyal)
- Guilherme Cassolato (@guicassolato)
- Jintao Zhang (@tao12345666333)
- Jongwoo Han (@jongwooo)
- Kevin Fan (@KevFan)
- Laura Fitzgerald (@laurafitzgerald)
- Mattia Lavacca (@mlavacca)
- Narasimha Murthy (@murthy95)
- Navendu Pottekkat (@pottekkat)
- Nick Young (@youngnick)
- Patryk Małek (@pmalek)
- Priyanka Saggu (@Priyankasaggu11929)
- Rob Scott (@robscott)
- Shane Utt (@shaneutt)
- Tarun Duhan (@tdn21)
- Vandit Singh (@Vandit1604)
- Yashvardhan Kukreja (@yashvardhan-kukreja)

Get Involved

- Contribute code, ideas, and expertise.
- Spread the word: Star, share, blog!

sigs.k8s.io/gwctl

Thank you!

Gateway API survey

#sig-network-gateway-api

https://github.com/kubernetes-sigs/gateway-api

https://github.com/kubernetes-sigs/gwctl