
KubeCon
CloudNativeCon
North America 2022
BUILDING FOR THE ROAD AHEAD
DETROIT 2022
TiKV
TiKV
HELM

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

SIG-NETWORK:
Intro and Deep Dive

Andrew Stoycos, Red Hat
Bowei Du, Google
Rob Scott, Google
Surya Seetharaman, Red Hat

Outline

- Networking APIs
  - Service, Endpoint(Slice), Ingress, Gateway API, DNS - Rob Scott
  - NetworkPolicy, Admin Network Policy, Multi Network - Surya
- Networking Components (Kube-Proxy, KPNG) - Andrew
- Features in development - Bowei
  - Topology
  - Terminating Endpoints
  - NetworkPolicy Status
  - InternalTrafficPolicy

Networking APIs

Services

- Enables grouping Pods together and exposing as a network Service
- Services are assigned IP address(es) they can be reached on
- Requests to those addresses will be routed to one of the associated Pods (endpoints)

```yaml
apiVersion: v1
kind: Service
metadata:
  name: store
spec:
  selector:
    app: store
  ports:
  - name: tcp
    protocol: TCP
    port: 80
    targetPort: 8080
```

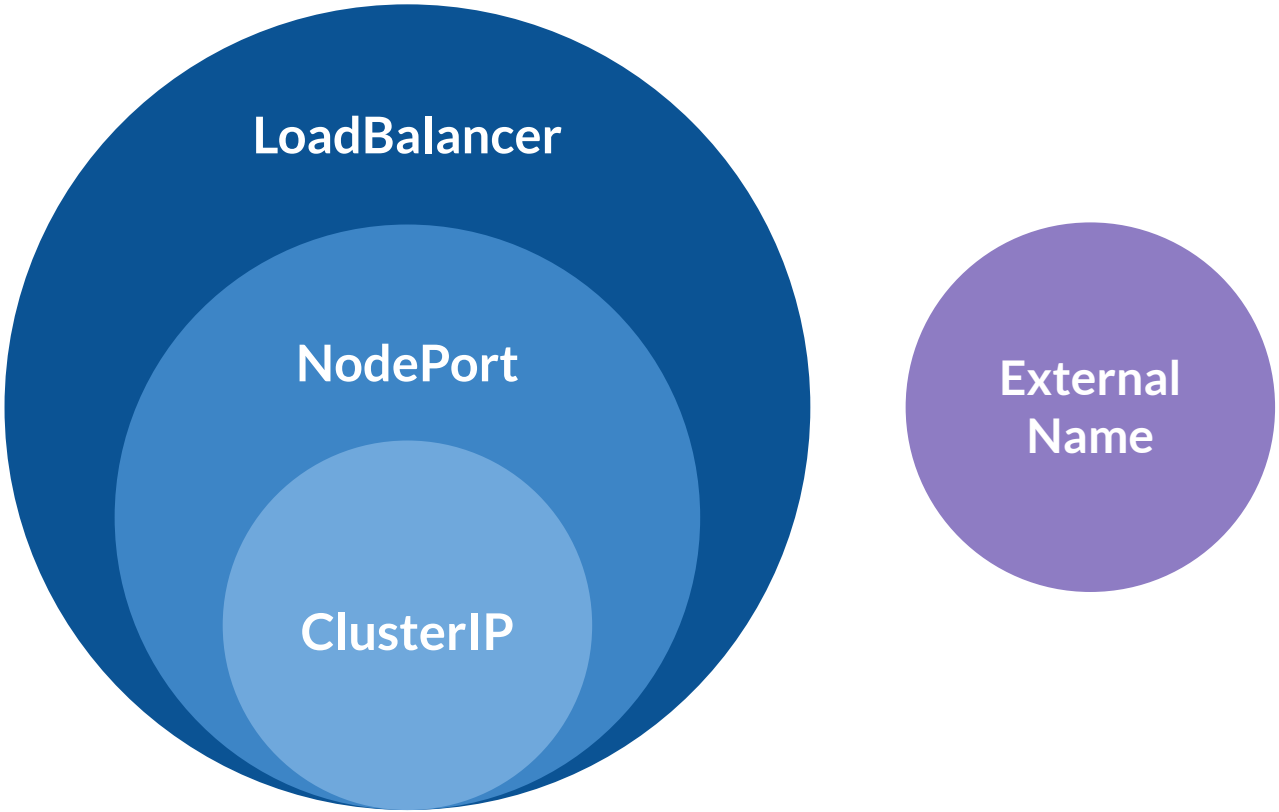
Services

- Enables grouping Pods together and exposing as a network Service
- Services are assigned IP address(es) they can be reached on
- Requests to those addresses will be routed to one of the associated Pods (endpoints)

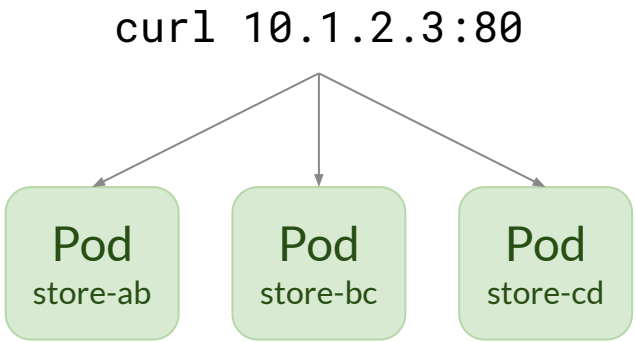
```yaml
apiVersion: v1
kind: Service
metadata:
  name: store
spec:
  selector:
    app: store
  ports:
  - name: tcp
    protocol: TCP
    port: 80
    targetPort: 8080
```

Service Types

- ClusterIP
- NodePort
- LoadBalancer
- ExternalName



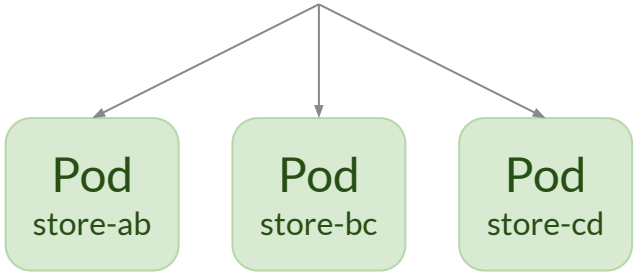
Reaching a Service




```yaml
apiVersion: v1
kind: Service
metadata:
  name: store
spec:
  clusterIP: 10.1.2.3
  selector:
    app: store
  ports:
  - name: tcp
    protocol: TCP
    port: 80
    targetPort: tcp
```

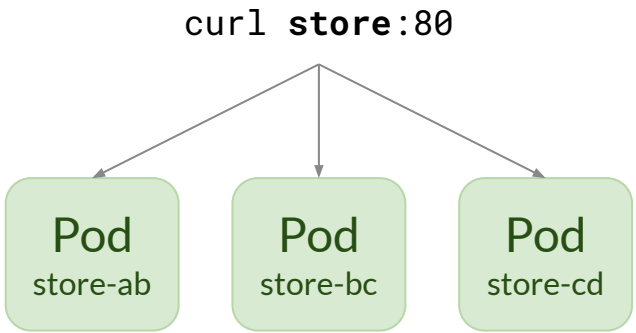
Reaching a Service

`curl` **`store.prod.svc.cluster.local`**`:80`




```yaml
apiVersion: v1
kind: Service
metadata:
  name: store
  namespace: prod
spec:
  clusterIP: 10.1.2.3
  selector:
    app: store
  ports:
  - name: tcp
    protocol: TCP
    port: 80
    targetPort: tcp
```

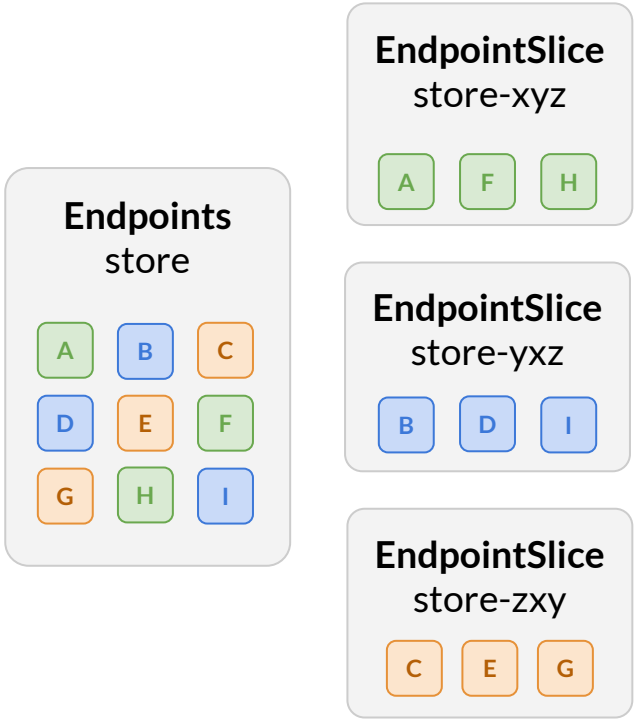
Reaching a Service




```yaml
apiVersion: v1
kind: Service
metadata:
  name: store
  namespace: prod
spec:
  clusterIP: 10.1.2.3
  selector:
    app: store
  ports:
  - name: tcp
    protocol: TCP
    port: 80
    targetPort: tcp
```

Endpoints and EndpointSlices

- Track IPs and Ports for Pods backing a Service
- Endpoints limited to 1000 Pods per Service
- EndpointSlices are sharded Endpoints, much more scalable
- New features enabled by EndpointSlices:
  - Dual Stack
  - Topology
  - Terminating Endpoints



Gateway and Ingress

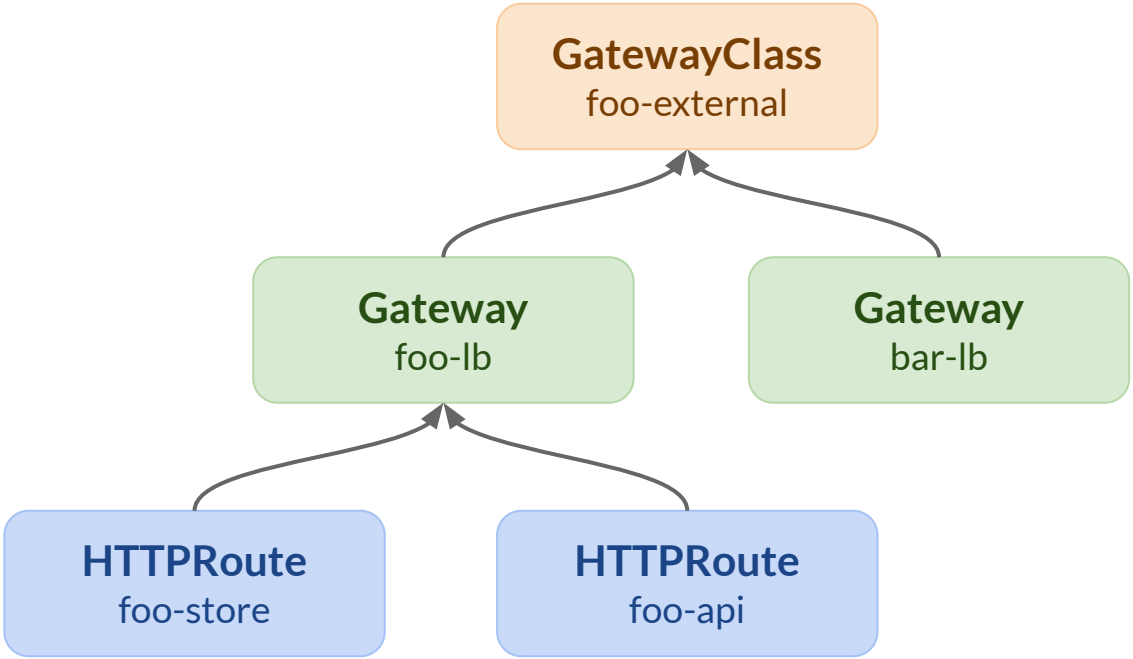
The Ingress API

- Host and Path Matching
- Forward to Service
- TLS Configuration
- Stable for 5 years
- **Simple and broadly implementable**

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: minimal-ingress
spec:
  ingressClassName: nginx-example
  rules:
  - http:
      paths:
      - path: /testpath
        pathType: Prefix
        backend:
          service:
            name: test
            port:
              number: 80
```

Gateway API

- Next generation of Kubernetes routing and load balancing APIs
- Designed to be expressive and extensible
- Role oriented resource model
- 15 implementations
- Graduated to beta in July



Simple Path Match

Ingress

```yaml
ingressClassName: nginx
rules:
- http:
    paths:
    - path: /login
      pathType: Prefix
      backend:
        service:
          name: auth-svc
          port:
            number: 8080
```

HTTPRoute

```yaml
parentRefs:
- name: nginx
rules:
- matches:
  - path:
      type: PathPrefix
      value: /login
  backendRefs:
  - name: auth-svc
    port: 8080
```

Get Involved

- Focus Areas: Mesh, L4, L7 -> GA, Conformance
- Weekly community meetings on Mondays
- Weekly GAMMA meetings on Tuesdays
- Contributors from all backgrounds welcome

gateway-api.sigs.k8s.io

#sig-network-gateway-api

kubernetes-sigs/gateway-api

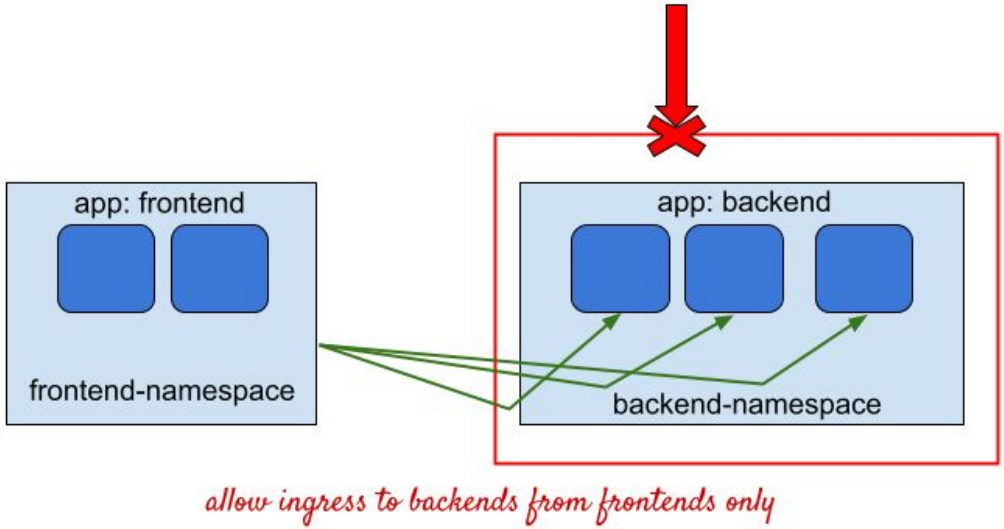
Network Policy

Contributions welcome! Stable for over 5 years.

network-policy-api && https://kubernetes.io/docs/concepts/services-networking/network-policies/

Network Policy (NP)

- How can app owners control traffic to/from their workloads?
    - example; backends can get traffic only from frontends, databases can only get traffic from backends etc..



Network Policy (NP)

- How can app owners control traffic to/from their workloads?
    - example; backends can get traffic only from frontends, databases can only get traffic from backends etc..
- An API that let's users define simple ingress/egress rules
- API design is implicit in nature
- Network policy peers
    - pod, namespace, ipBlock

```yaml
---
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-ingress-to-backend-from-frontend
  namespace: foo
spec:
  podSelector:
    matchLabels:
      app: backend
  policyTypes:
    - Ingress
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          app: frontend
```

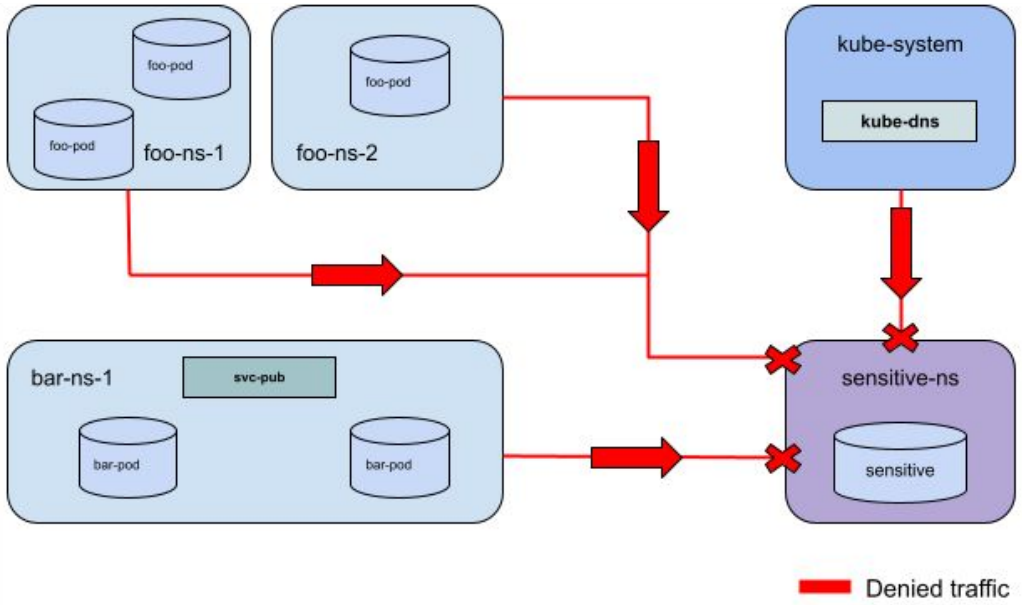
Admin Network Policy

Contributions welcome! Under active development!

https://github.com/kubernetes-sigs/network-policy-api && https://network-policy-api.sigs.k8s.io/

Admin Network Policy (ANP)

- Network Policies were designed for app owners…
- How can admins enforce policies cluster-wide??



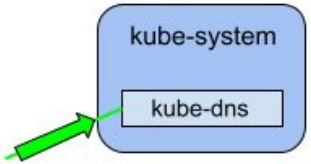
Admin Network Policy (ANP)

- Network Policies were designed for app owners…
- How can admins enforce policies cluster-wide??
- Cluster-scoped policy API
  - AdminNetworkPolicy
  - BaselineAdminNetworkPolicy
- API design is explicit in nature

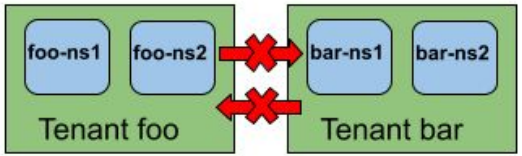
```yaml
apiVersion: policy.networking.k8s.io/v1alpha1
kind: AdminNetworkPolicy
metadata:
  name: cluster-wide-deny-example
spec:
  priority: 1
  subject:
    namespaces:
      matchLabels:
        kubernetes.io/metadata.name: sensitive-ns
  ingress:
  - name: "default-deny-to-sensitive-ns"
    action: "Deny"
    from:
    - namespaces:
        related: "NotSelf"
```

Admin Network Policy (ANP)

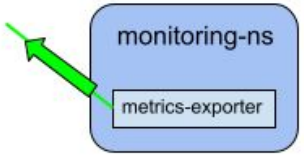
- Network Policies were designed for app owners…
- How can admins enforce policies cluster-wide??
- Cluster-scoped policy API
  - AdminNetworkPolicy
  - BaselineAdminNetworkPolicy
- API design is explicit in nature
- v1alpha1 supports east-west traffic
- TBD: north-south traffic support
- Next Steps: Implementations!



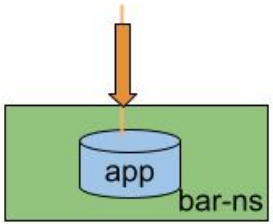
always ingress to dns namespace



isolate multi-tenants



always egress from monitoring namespace



explicitly delegate to network policy rules in an app namespace

Get Involved

- Focus Areas: Network Policies, Admin Network Policies
- Bi-Weekly community meetings on Monday at 4pm EDT/1pm PDT
- All are Welcome!

#sig-network-policy-api

kubernetes-sigs/network-policy-api

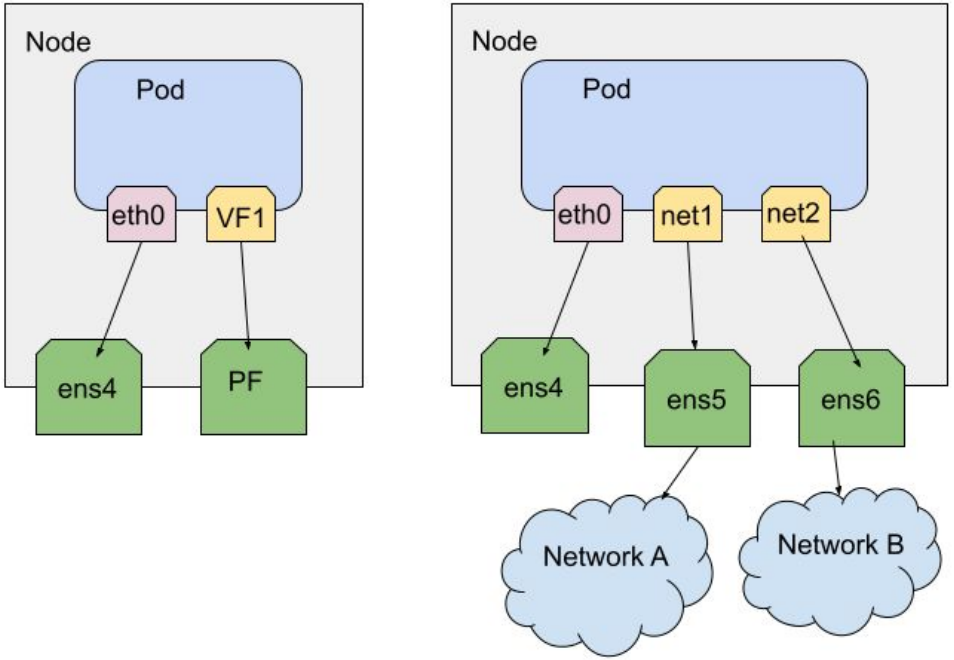
Multi Network

Contributions welcome! Ongoing design! New SIG!

https://docs.google.com/document/d/1ztx9TOQ9Hiyj9PG9aPv6jyDLhe_FB7haV_yjJIcb-0Y/edit#heading=h.5t6qeinm4208

Multi-Network

- Apps might need to access different isolated networks via performance oriented interfaces like SR-IOV
- Represent more complex networking solutions via an API
- In early stages (design phase)



Get Involved

KubeCon CloudNativeCon North America 2022

- Focus Areas: Multi Network (intro slide deck)
- Bi-Weekly community meetings on Wednesday at 8am PST/ 5pm CEST
- Welcome!

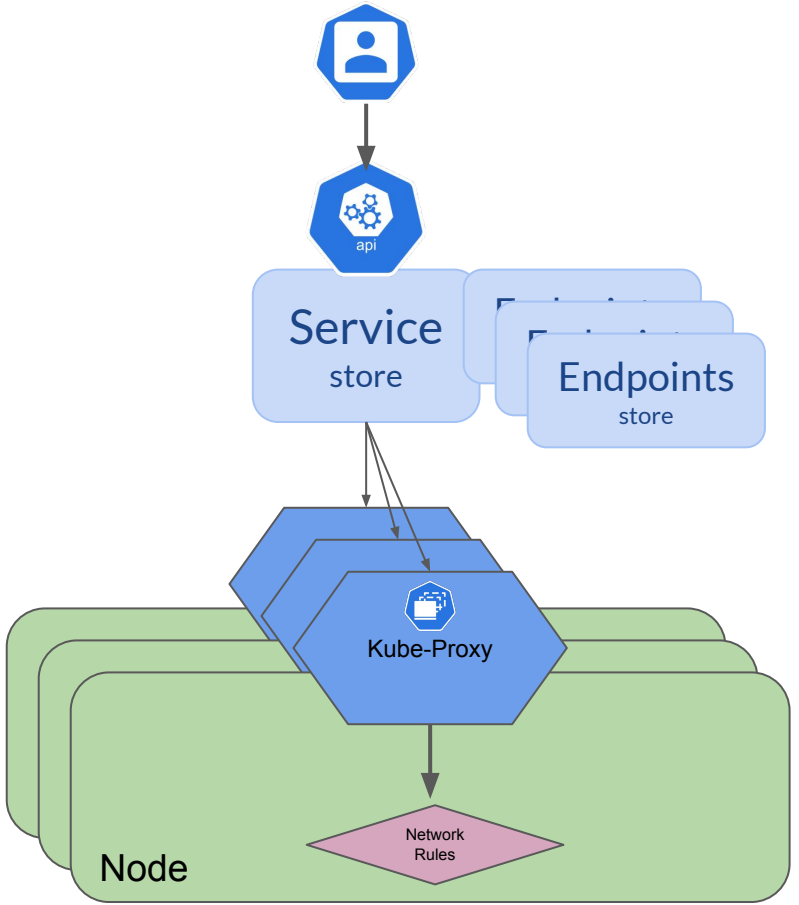
#sig-network-multi-network

Networking Components

Kube-Proxy

Kube-Proxy

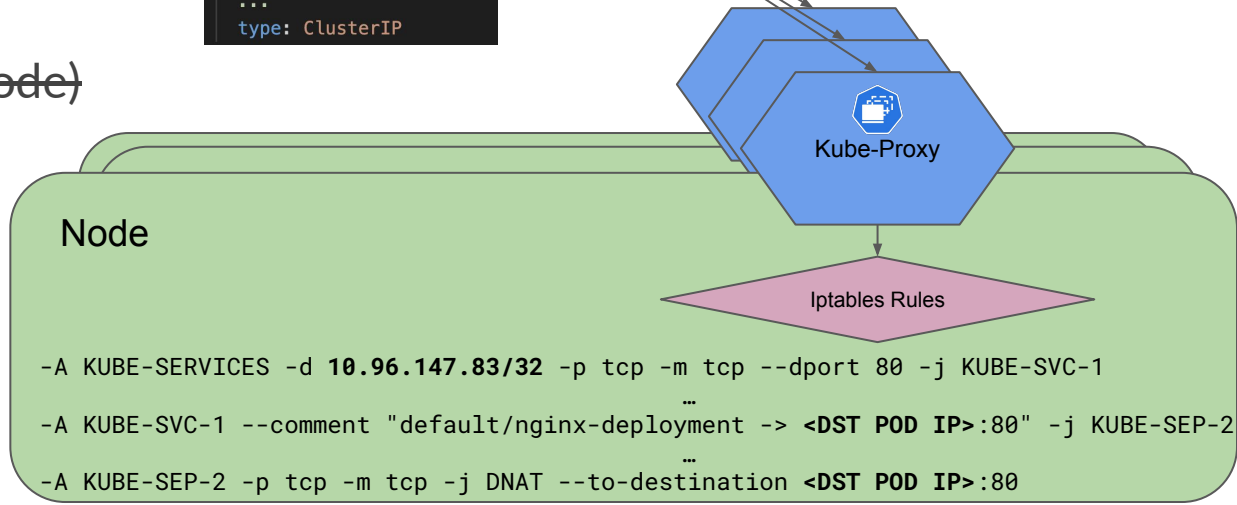
- We talked about some great K8s Networking APIs including **Services** and **Endpoints**
- Kube-Proxy implements the Services API
- Node agent which resolves K8s Networking Objects into datapath rules



Kube-Proxy

- Implemented in core Kubernetes
- Can program datapath rules in three different modes
  - iptables (default)
  - ipvs
  - ~~userspace ("legacy" mode)~~

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-deployment
  namespace: default
  ...
spec:
  clusterIP: 10.96.147.83
  ...
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  ...
  type: ClusterIP
```



BUILDING FOR THE ROAD AHEAD

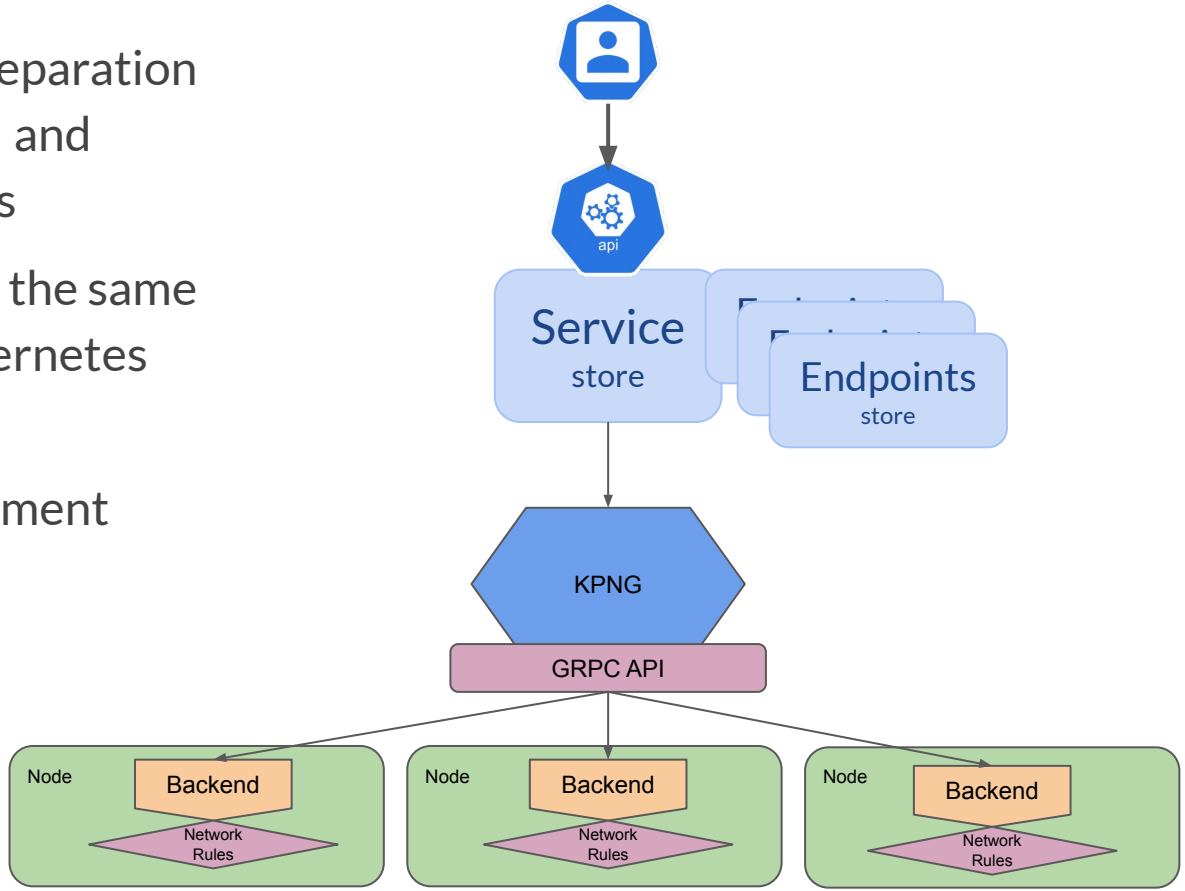
DETROIT 2022

KPNG

(Kube-Proxy Next Generation)

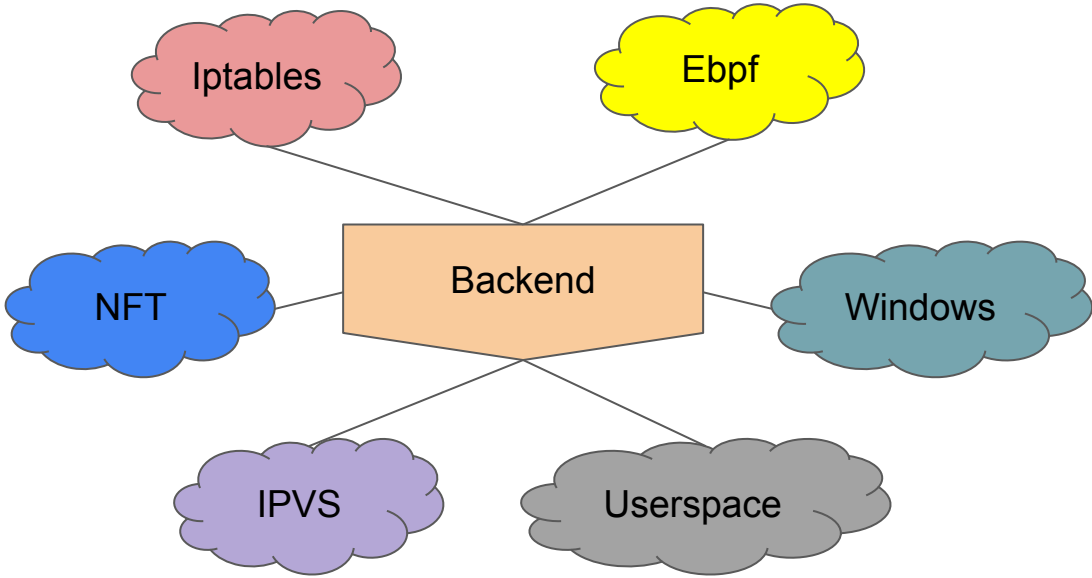
KPNG

- Provides a fundamental separation between the Services API and Backend Implementations
- Allows backends to reuse the same bits which watch the kubernetes API
- Extremely flexible deployment model



KPNG

- Current Backends
  - Iptables
  - NFT
  - Ipvs
  - Userspace
  - Windows
  - Ebpf (POC)
- More to come from Viewers like You!



Networking Components: Get Involved

- Now that you know a bit more….Get involved!
  - So You Want To Write A Service Proxy…
  - Kube-Proxy good-first-issues
  - KPNG good-first-issues

Features in Development

Topology Aware Hints

Pods

Zone A

A A A A

Zone B

B B B

Zone C

C C

EndpointSlices

Original

For: All Zones

A A A
A B B
B C C

Allocate proportional endpoints to each zone

Topology Hints

For: Zone A

A A A

For: Zone B

B B B

For: Zone C

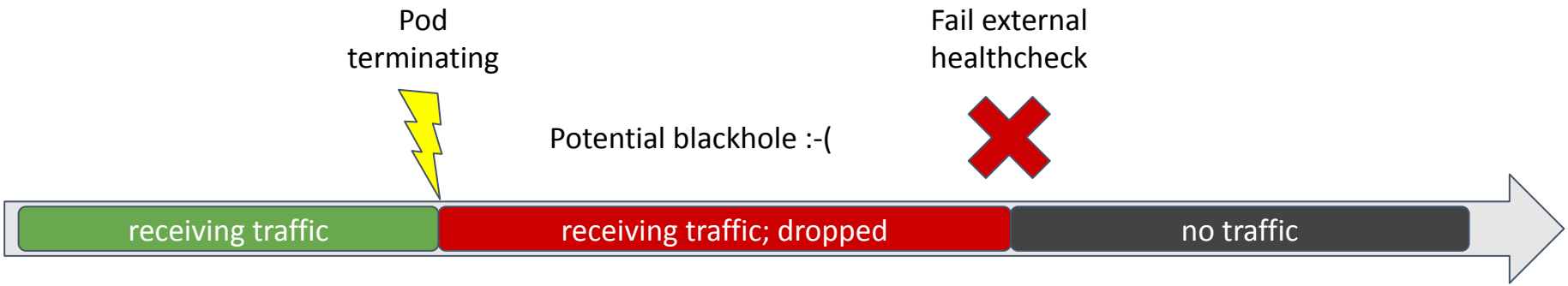
C C A

Terminating Endpoints

Tracking terminating endpoints (KEP, KEP)

Gracefully keep sending traffic to an `ExternalTrafficPolicy=local` Pods to include those that are terminating. This avoids a **blackhole** between when the Pods terminate and when the LB notices the Pods are shutting down.

Adds “terminating” state to EndpointSlice.

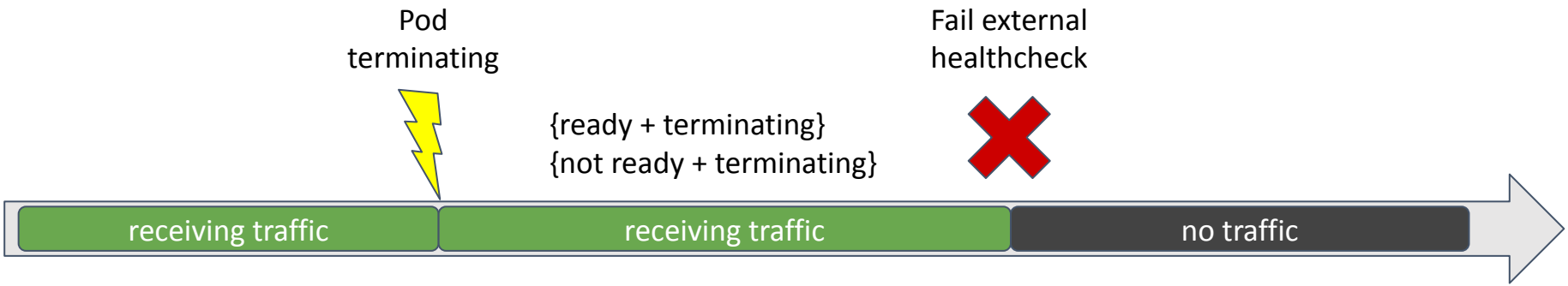


Terminating Endpoints

Tracking terminating endpoints (KEP, KEP)

Gracefully keep sending traffic to an `ExternalTrafficPolicy=local` Pods to include those that are terminating. This avoids a **blackhole** between when the Pods terminate and when the LB notices the Pods are shutting down.

Adds “terminating” state to EndpointSlice.



NetworkPolicy Status

NetworkPolicy now has a Status and Conditions fields.

This allows implementations to signal state about your policy:

- Optional feature support such as portRanges.

```
apiVersion: v1
kind: NetworkPolicy
metadata:
  name: my-policy
spec:
  ...
status:
  conditions:
  - ...
```

InternalTrafficPolicy

Makes your service route only to endpoints local to the Node for traffic sent to the Service ClusterIP.

Typical use case: Service backed by a Daemonset for a logging service with traffic that you never want to traverse outside of the node.

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app.kubernetes.io/name: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
  internalTrafficPolicy: Local
```

Questions?