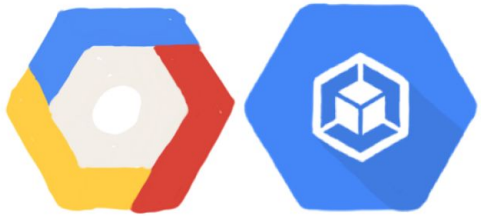
SECURING CLUSTERS WITH

Kubernetes Network Policies

@ahmetb

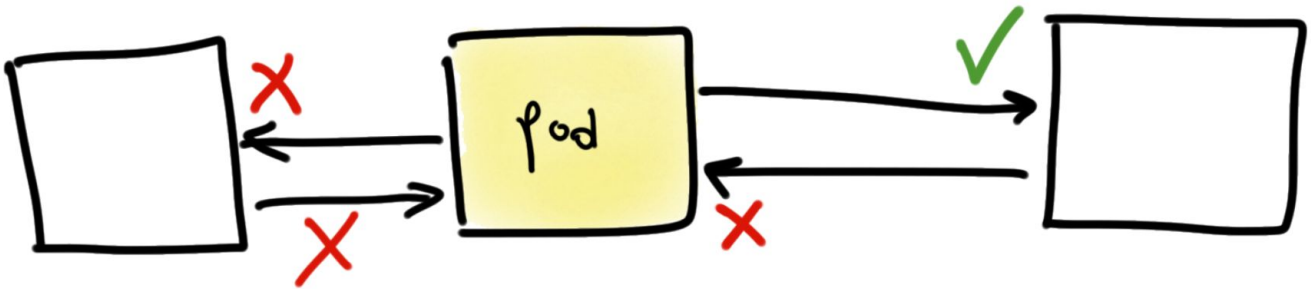
software engineer @ Google Cloud



AGENDA

1. what are Network Policies
2. how to write them
3. how do they work
4. go home and use them

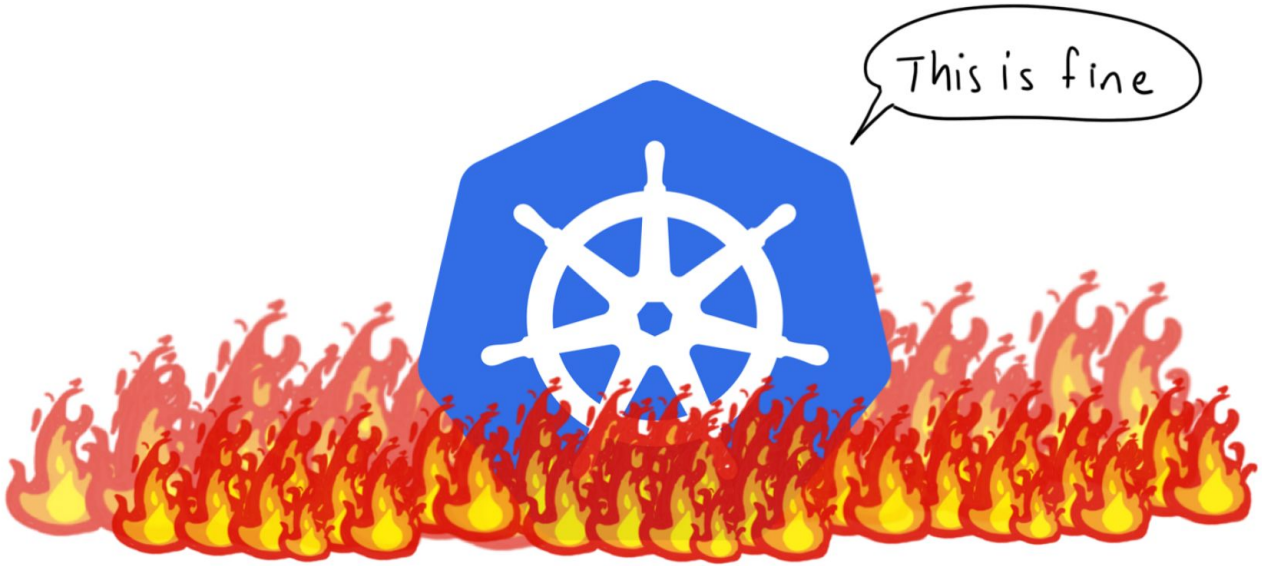
network policies control traffic from/to pods



STRAW POLL

Who is using Network Policies?

Kubernetes clusters without Network Policies



pre-work ⟶ late 2015

alpha ⟶ v1.2 (March 2016)

beta ⟶ v1.3 (July 2016)

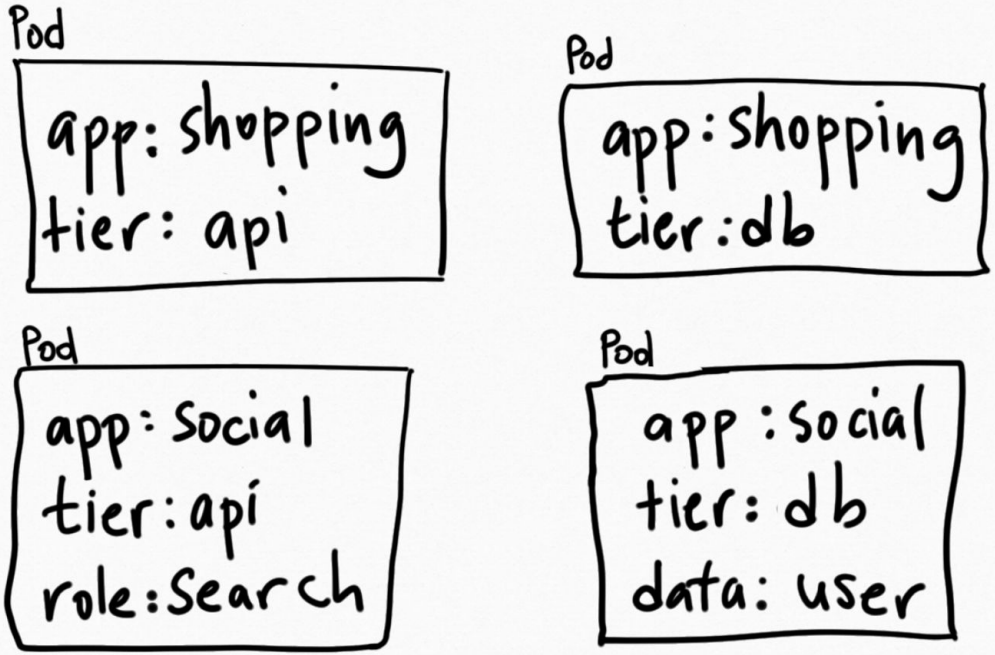
stable ⟶ v1.7 (June 2017)

---

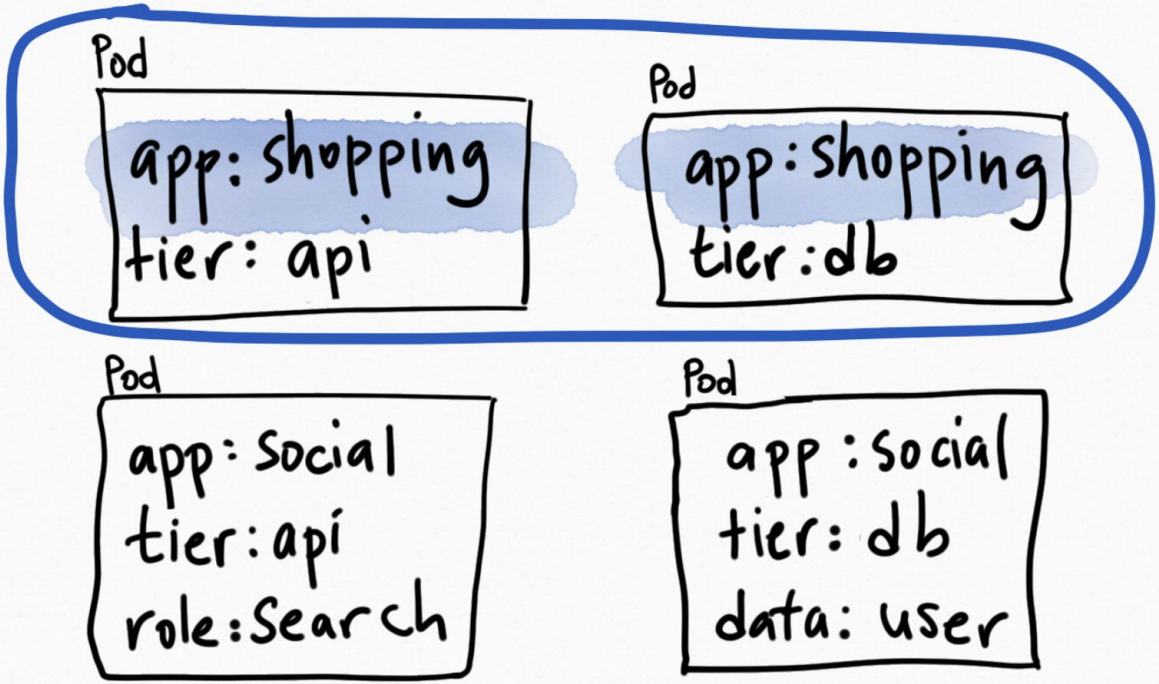
thanks to: Casey Davenport (Tigera), Christopher Luciano (IBM), Dan Winship (Red Hat)

writing network policies

but first, you need to know: **labels**



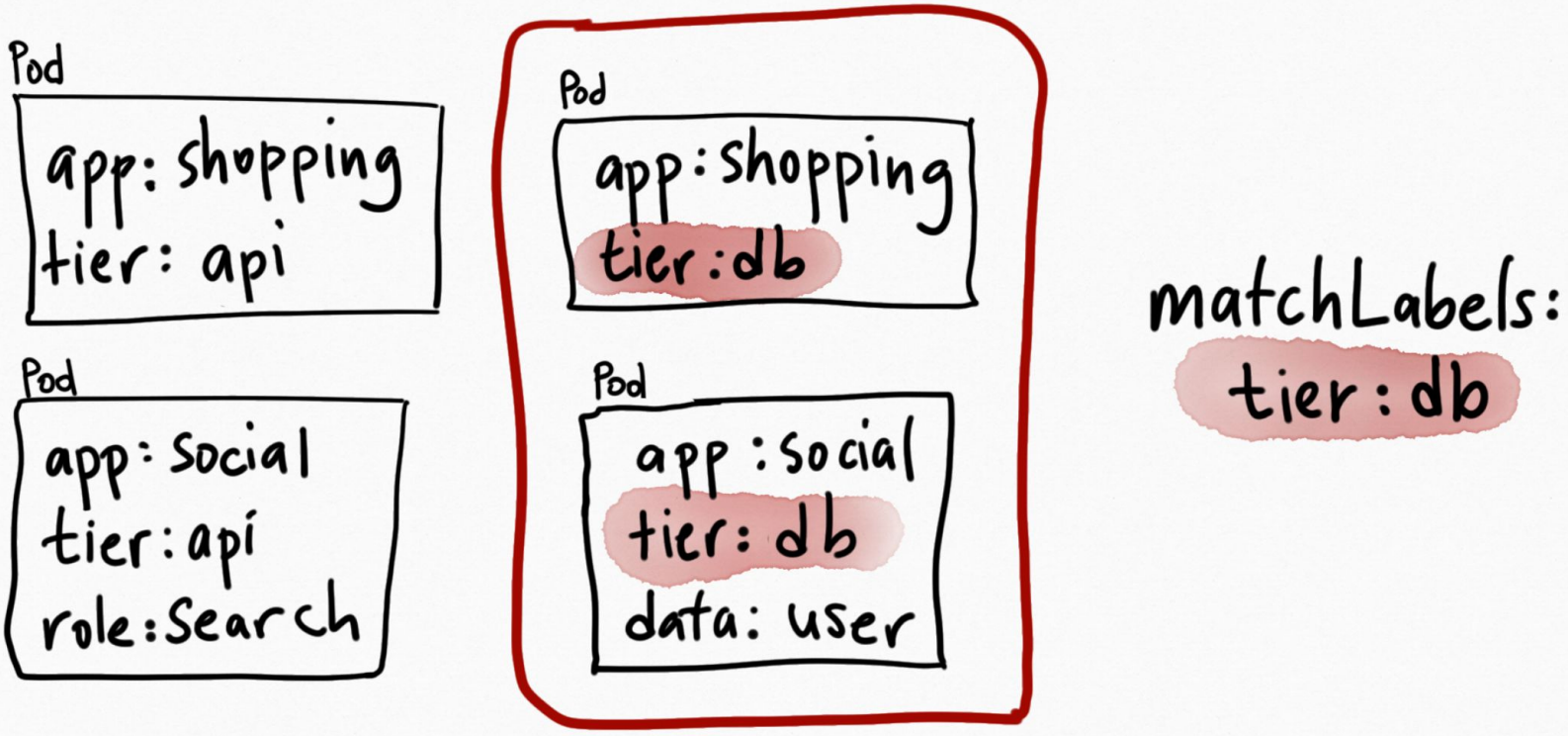
label selectors



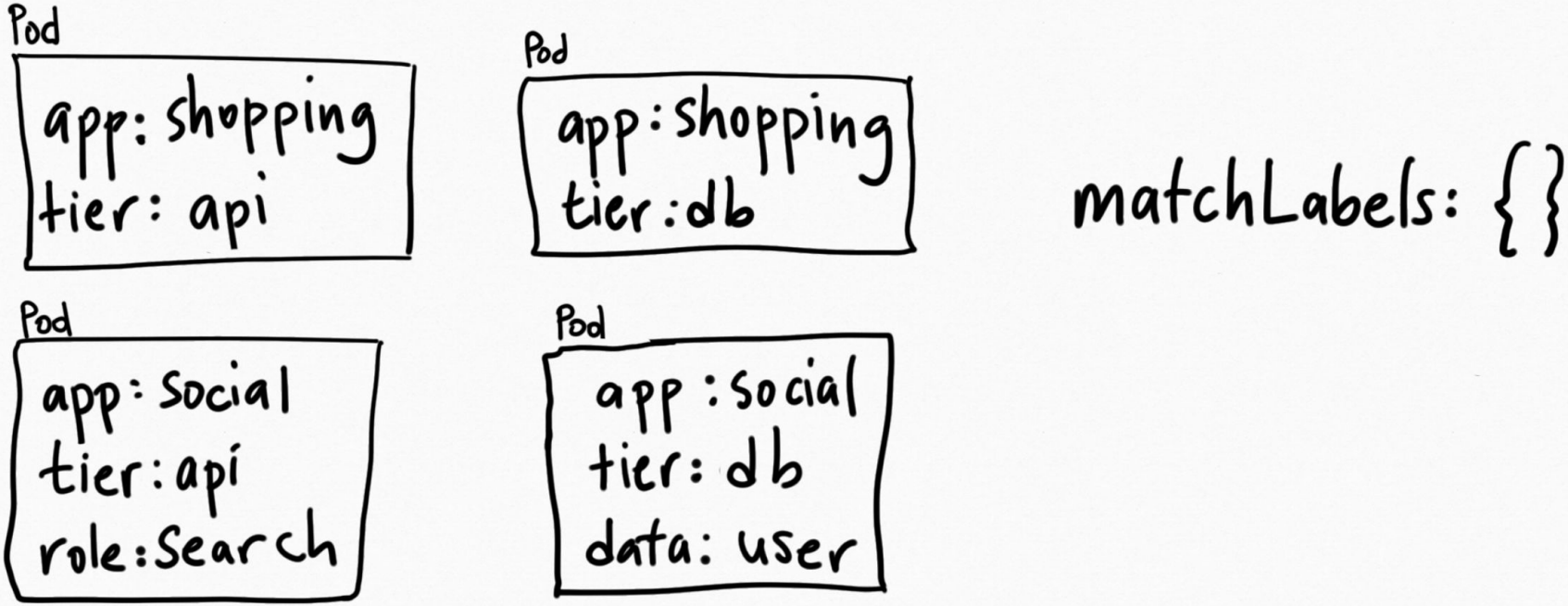
matchLabels:

app: shopping

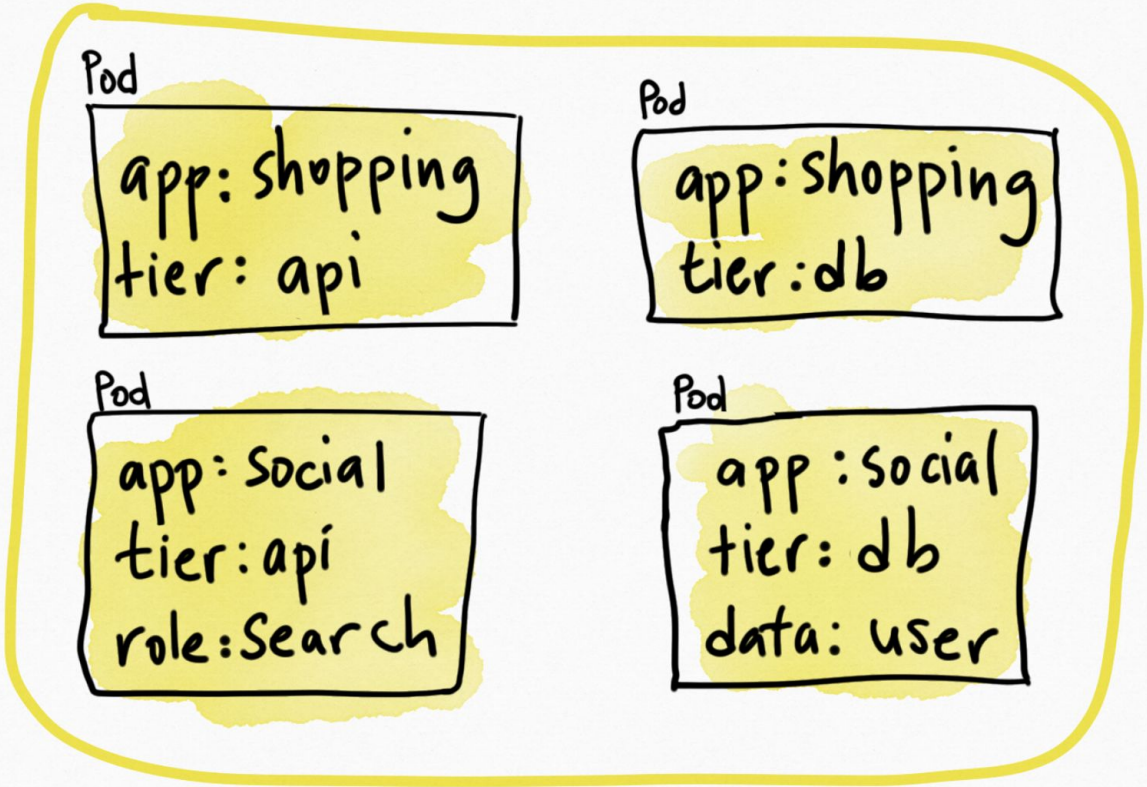
label selectors



empty label selector



empty label selector



matchLabels: {}

anatomy of a Network Policy

① which pods does it apply to

anatomy of a Network Policy

1. which pods does it apply to
2. for which direction?
   - Ingress: → Pod
   - Egress: Pod →

anatomy of a Network Policy

1. which pods does it apply to
2. for which direction?
   - Ingress: → Pod
   - Egress: Pod →
3. rules for allowing:
   - Ingress → who can connect to this Pod?
   - Egress → where can this pod connect to?

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: ...
  namespace: ...
  ...
spec:
  podSelector: ...
  ingress:
  - ...
  - ...
  egress:
  - ...
  - ...
```

RULE ①

traffic is allowed
unless there is a Network Policy
selecting the pod

RULE ②

traffic is denied

if there are policies selecting the pod,

but none of them have any rules allowing it.

(RULE ① + ②)

You can only write rules
that allow traffic.

Denying all traffic by matching to it

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny-all
spec:
  podSelector: {}
  ingress: []
```

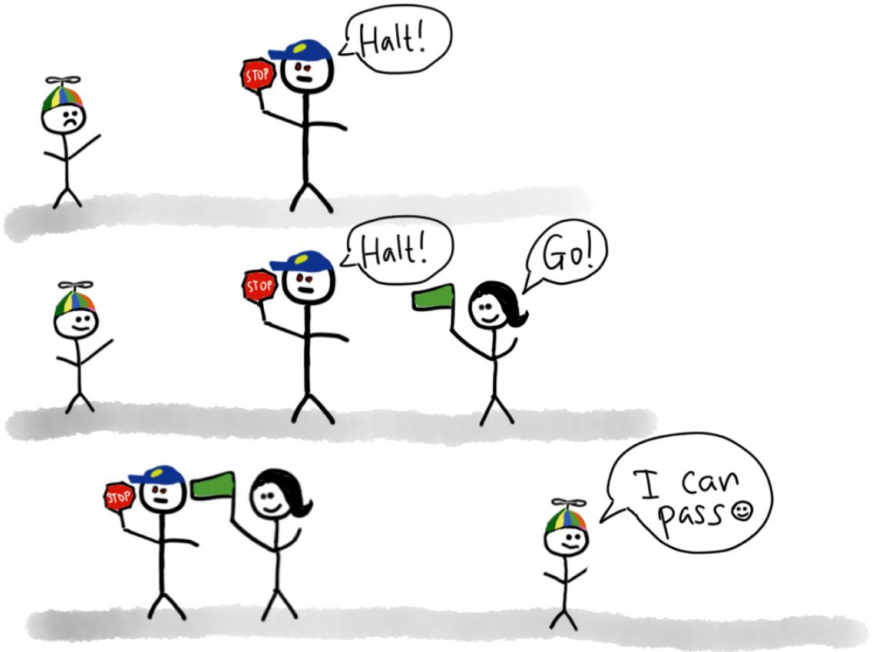
→ select all pods

→ empty array indicates nothing is whitelisted

RESULT: all ingress traffic is blocked by matching, and not allowing.

RULE 3

traffic is allowed if there is at least one policy allowing it



Example : allowing some traffic

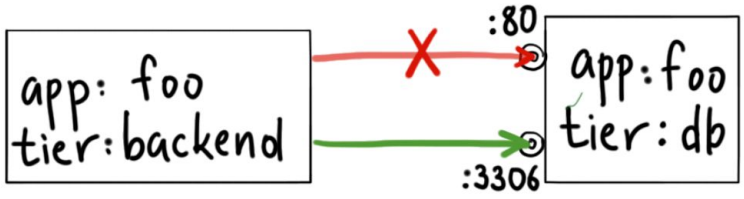
Example : allowing some traffic


```yaml
kind: NetworkPolicy
spec:
  podSelector:
    matchLabels:
      app: foo
      tier: db
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: foo
          tier: backend
```

for these pods

allow traffic from these

restricting port numbers




```
- from:
  - podSelector:
      matchLabels:
        app: foo
        tier: backend
  ports:
  - port: 3306
    protocol: TCP
```

Specifying port numbers

- Can't use Service ports in policies, yet.
- Use containerPort in the PodSpec.

RULE ④

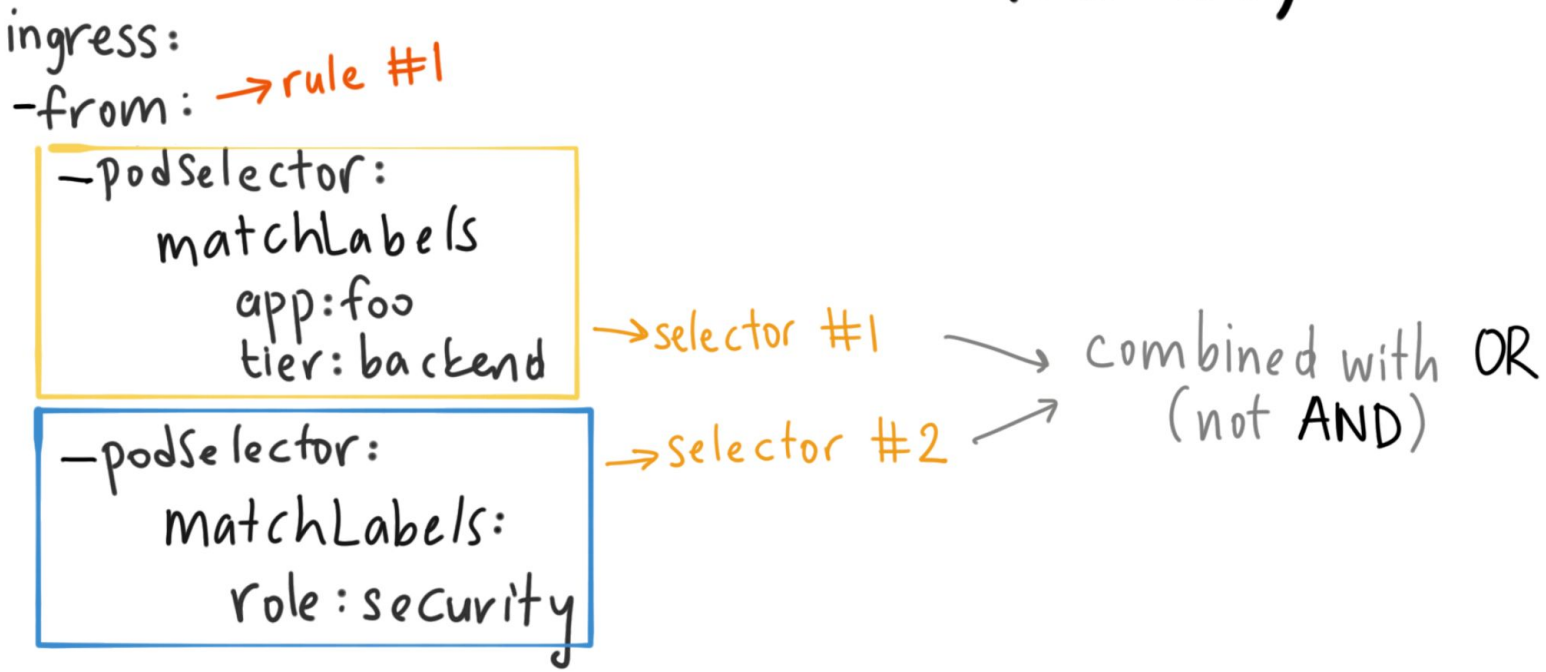
Policy rules are additive.

They are OR'ed with each other.

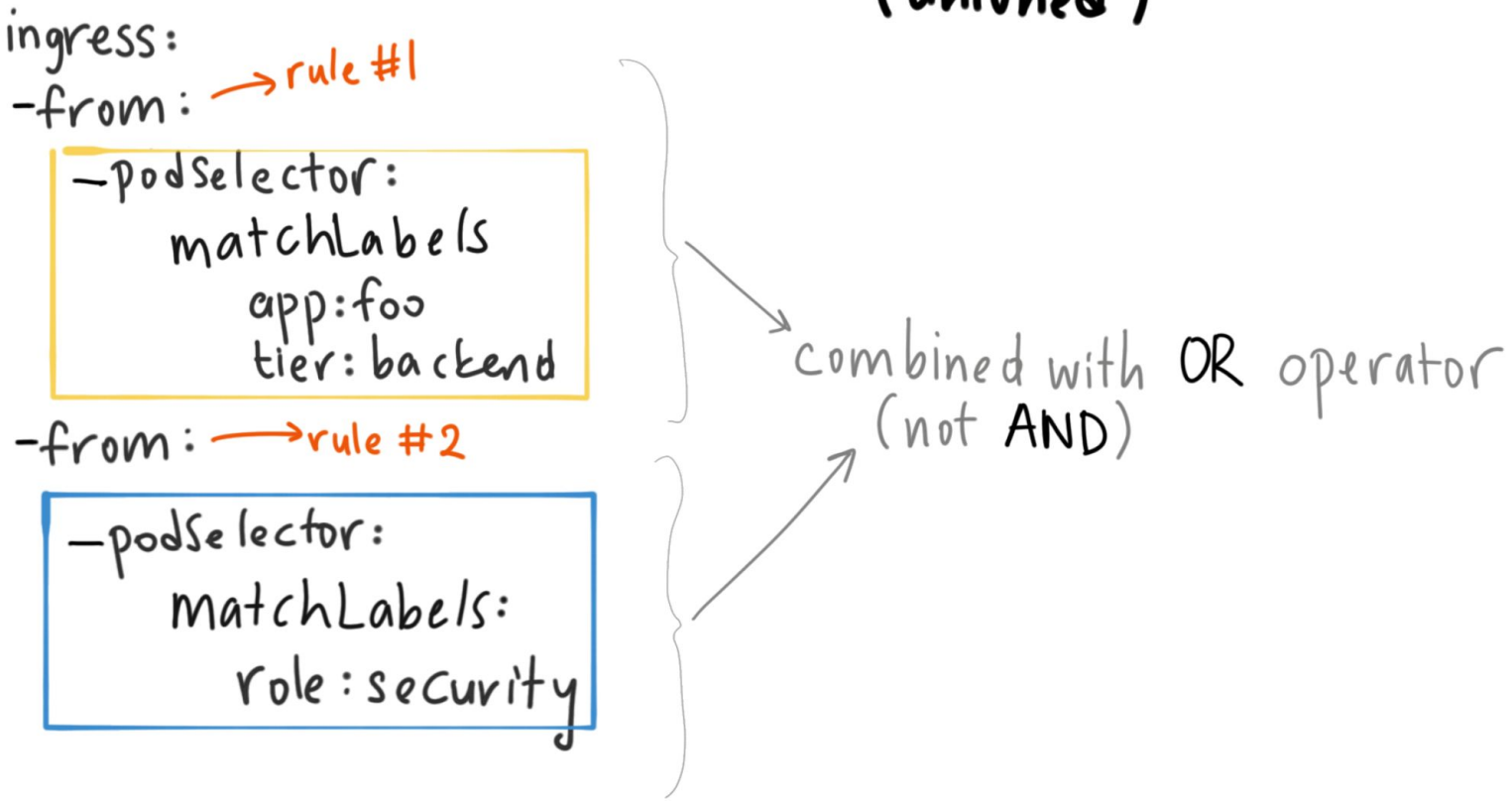
```
def allowed(traffic):
    return role1.allows(traffic) ||
           role2.allows(traffic) ||
           ...
           roleN.allows(traffic)
```

example

multiple selectors are combined (unioned)



multiple rules are combined, too (unioned)



Using empty selectors

"allow all monitoring pods to connect port 5000 of all pods."

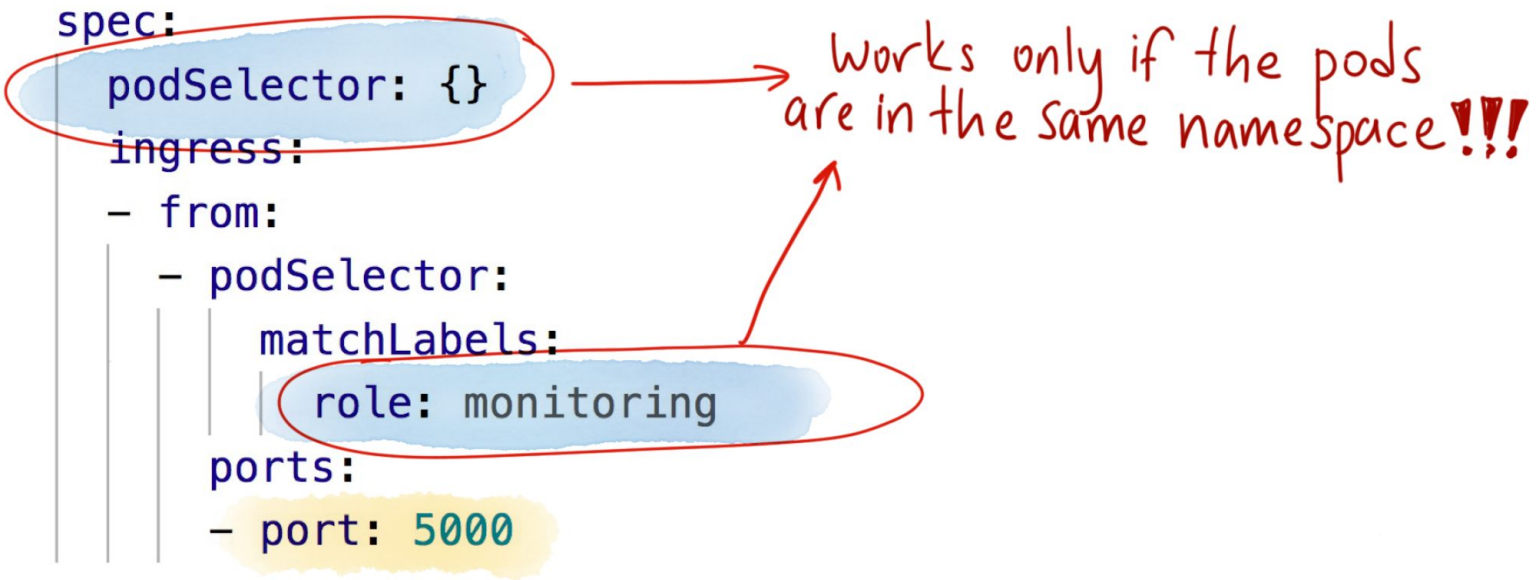
Using empty selectors

"allow all monitoring pods to connect port 5000 of all pods."

```
spec:
  podSelector: {}
  ingress:
  - from:
    - podSelector:
        matchLabels:
          role: monitoring
      ports:
      - port: 5000
```

Using empty selectors

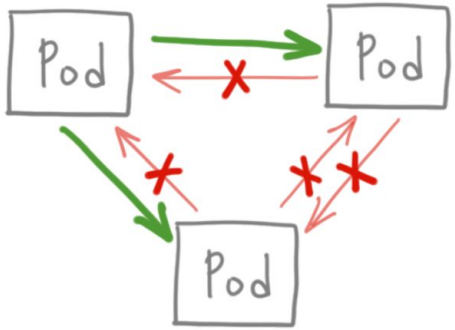
"allow all monitoring pods to connect port 5000 of all pods."



RULE ⑤

Network policies are scoped to the namespace they're deployed to.

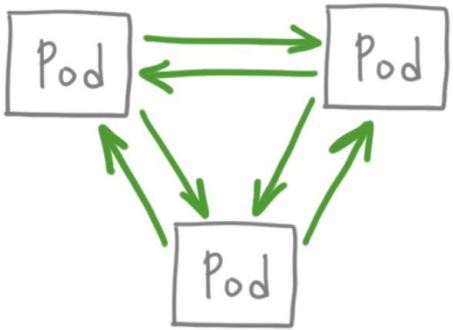
→ "spec.podSelector" does not select pods from other namespaces



namespace: foo

---

namespace: bar



(no policy enforcement)

allowing traffic from other namespaces

* `from.podSelector` matches only to pods in the current namespace
* need a new kind of selector to choose pods from other namespaces.

allowing traffic from other namespaces

- from.podSelector matches only to pods in the current namespace
- need a new kind of selector to choose pods from other namespaces.

namespaceSelector

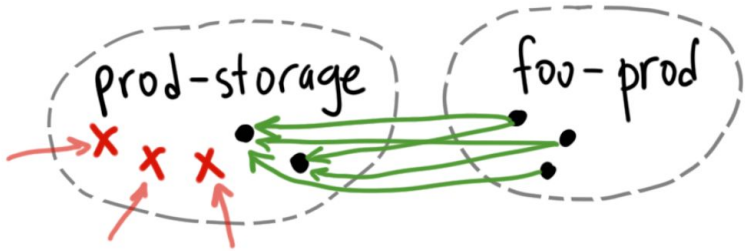
- like podSelector
- selects namespaces, using labels

who is labeling their namespaces?

allowing traffic from other namespaces

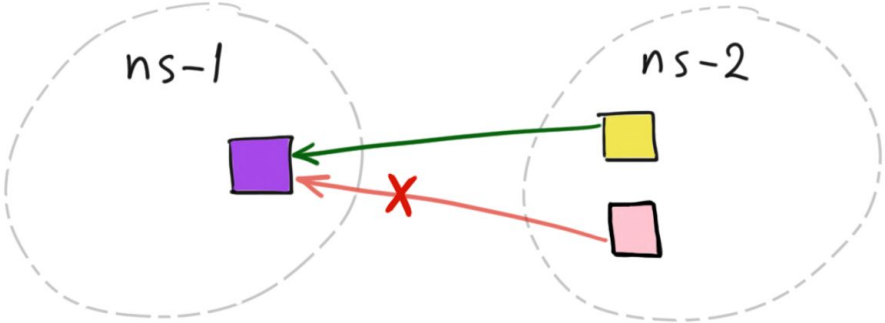
```yaml
metadata:
  namespace: prod-storage
  name: allow-prod-apps
spec:
  podSelector: {}
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          purpose: prod
```

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: foo-prod
  labels:
    purpose: prod
    product: foo
```



limitation

allowing some pods from other namespaces



(this is not possible today)

3 ways to specify where the traffic can
- come from 
- go to 

1. podSelector ✓
2. namespaceSelector ✓
3. ipBlock ?

the "ipBlock" selector

CIDR refresher	
10.0.0.0/8	10.x.x.x
192.1.2.0/24	192.1.2.x
0.0.0.0/0	x.x.x.x

allow 10.56.x.x

```yaml
from:
- ipBlock:
    cidr: 10.56.0.0/16
```

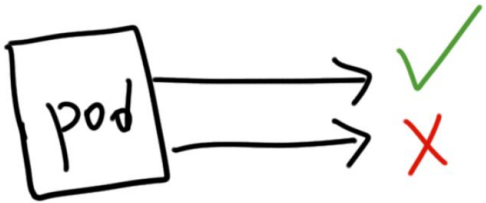
allow 10.x.x.x
but deny 10.11.12.x

```yaml
from:
- ipBlock:
    cidr: 10.0.0.0/8
    except:
    - 10.11.12.0/24
```

egress rules

almost identical to ingress rules

controls the traffic from selected pods



Denying all egress traffic

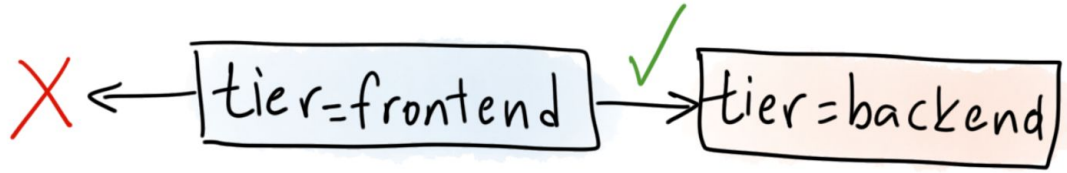
```yaml
kind: NetworkPolicy
metadata:
  name: default-deny-all-egress
spec:
  podSelector: {}
  policyTypes:
  - Egress
  egress: []
```

→ select all pods in the namespace

→ need this if you have egress rules

→ empty array = no rules.
nothing is whitelisted

allowing some egress traffic




```yaml
spec:
  podSelector:
    matchLabels:
      tier: frontend
  policyTypes:
  - Egress
  egress:
  - to:
    - podSelector:
        matchLabels:
          tier: backend
```

allowing some egress traffic

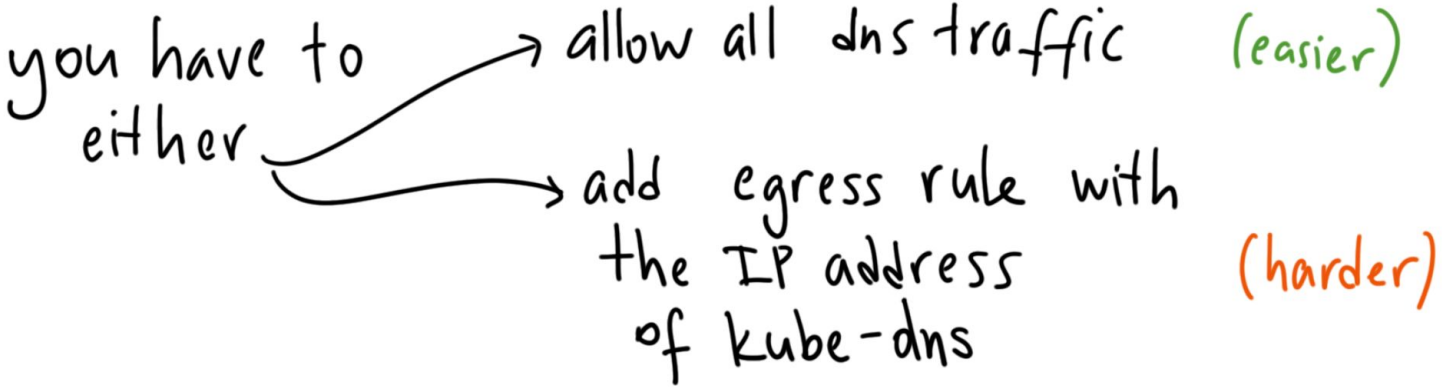

```yaml
spec:
  podSelector:
    matchLabels:
      tier: frontend
  policyTypes:
  - Egress
  egress:
  - to:
    - podSelector:
        matchLabels:
          tier: backend
```

(on frontend Pod)

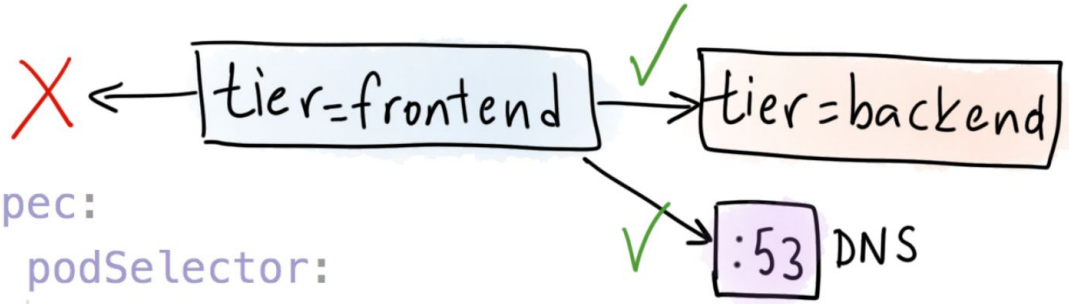
```
$ curl http://backend
cannot resolve host: 'backend'
```

????

egress policies actually block dns resolution



allowing some egress traffic & dns




```yaml
spec:
  podSelector:
    matchLabels:
      tier: frontend
  policyTypes:
  - Egress
  egress:
  - to:
    - podSelector:
        matchLabels:
          tier: backend
```

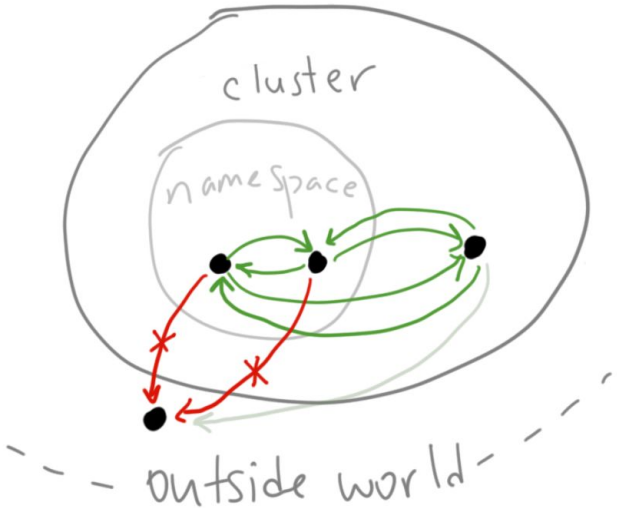
rule #1

rule #2

```yaml
  - ports:
    - port: 53
      protocol: "UDP"
    - port: 53
      protocol: "TCP"
```

more fun with egress

blocking external traffic




```yaml
metadata:
  name: allow-local-egress
spec:
  podSelector: {}
  policyTypes:
  - Egress
  egress:
  - to:
    - namespaceSelector: {}
```

enforce policy for all pods in this namespace

all namespaces (with all their pods)

how are

network policies

enforced

surprise!

without a networking plugin
that enforces policies,
NetworkPolicy objects are
Silently ignored

install a plugin like Calico, WeaveNet or Romana
(not an endorsement)

GKE network policy (beta)

powered by Calico

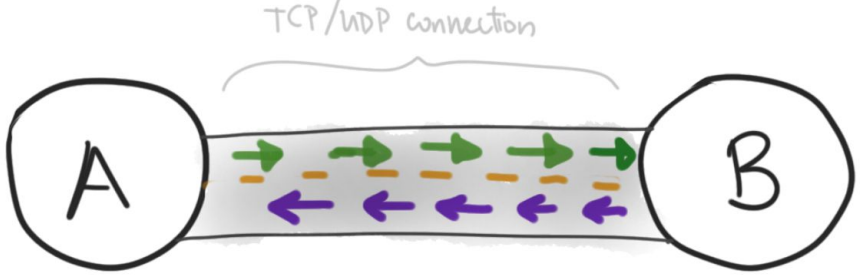
```
$ gcloud beta container clusters create
    --enable-network-policy
```

→ installs Calico network plugin
→ in partnership with Tigera
→ releases managed by Google

→ cloud.google.com/kubernetes-engine

Connections are duplex (2-way)

once A connects B, B can send data to A, on the same connection.

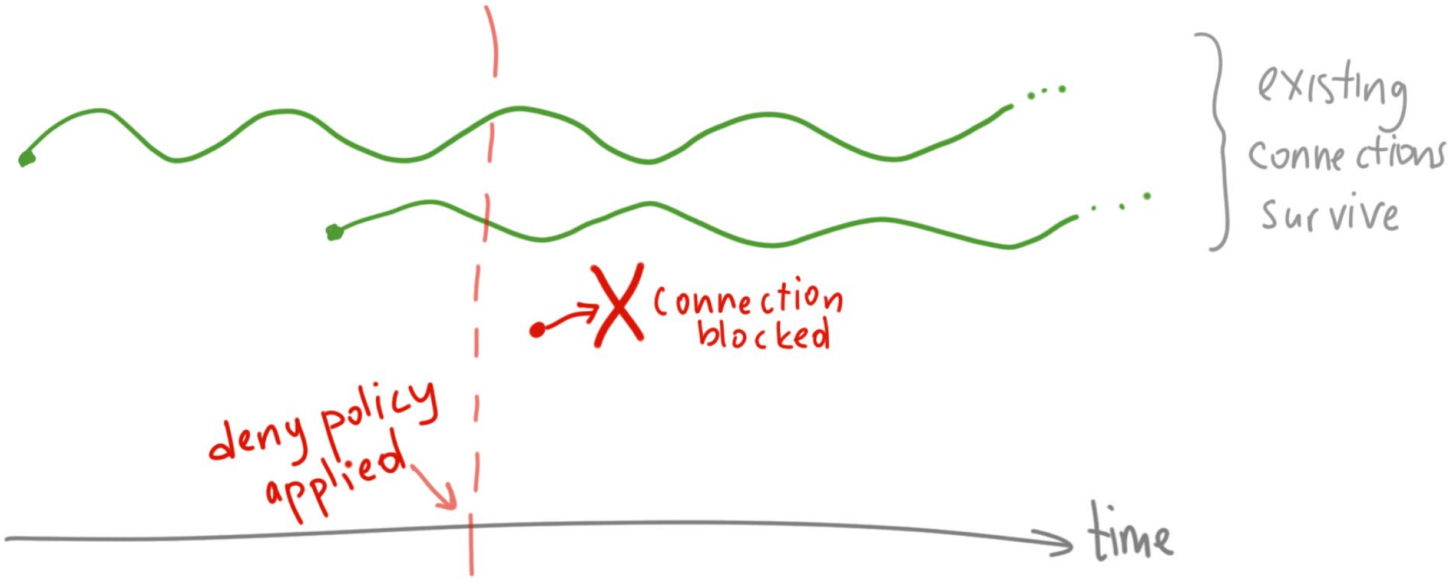


(does not mean B can connect to A)

network policy is a connection filter

→ does not apply to packets

→ does not terminate established connections



Network policies add minimal latency

→ depends on the network plugin
- iptables vs overlay
- implementation, caching differences

→ can depend on number of policies deployed

→ sub-1ms. overhead [1]

[1] http://blog.kubernetes.io/2016/09/high-performance-network-policies-kubernetes.html

best practices

use "default-deny-all" rules

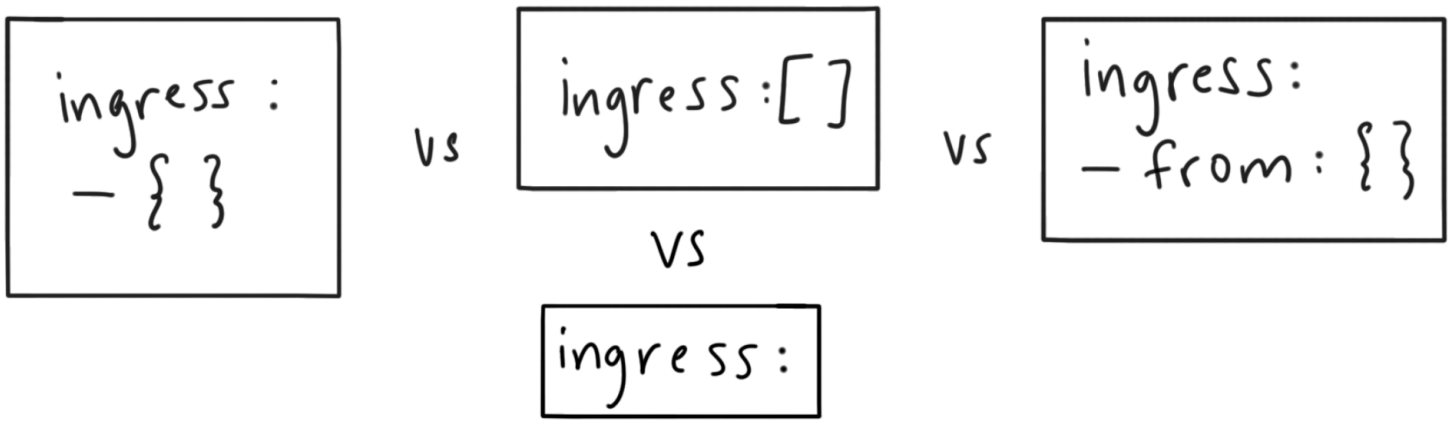
first block all ingress/egress in a namespace

then start whitelisting for each application

understand rule evaluation

* rules are OR'ed (not ANDed)
  → additivity may cause unexpected results

* be explicit about empty vs null fields

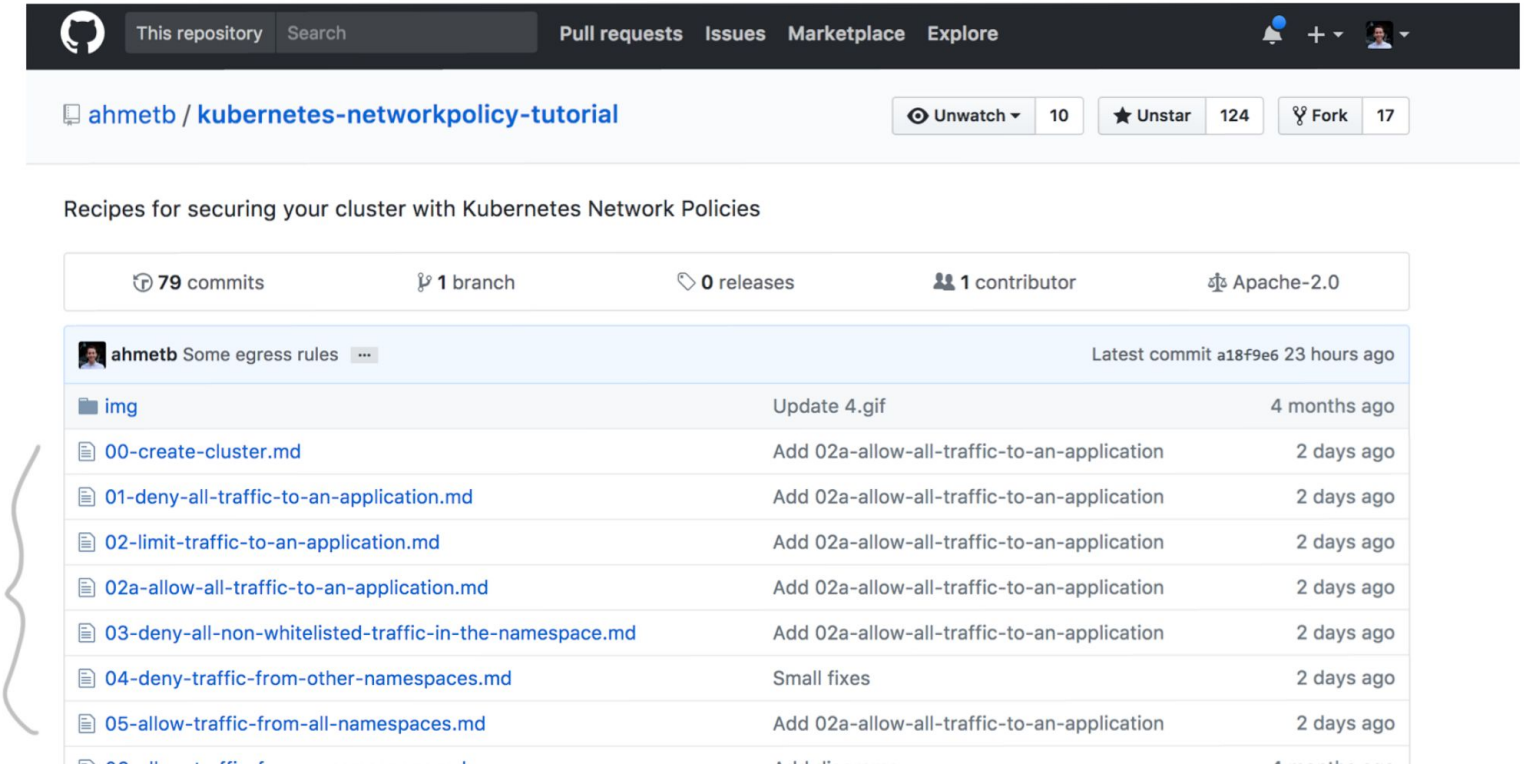


test your policies

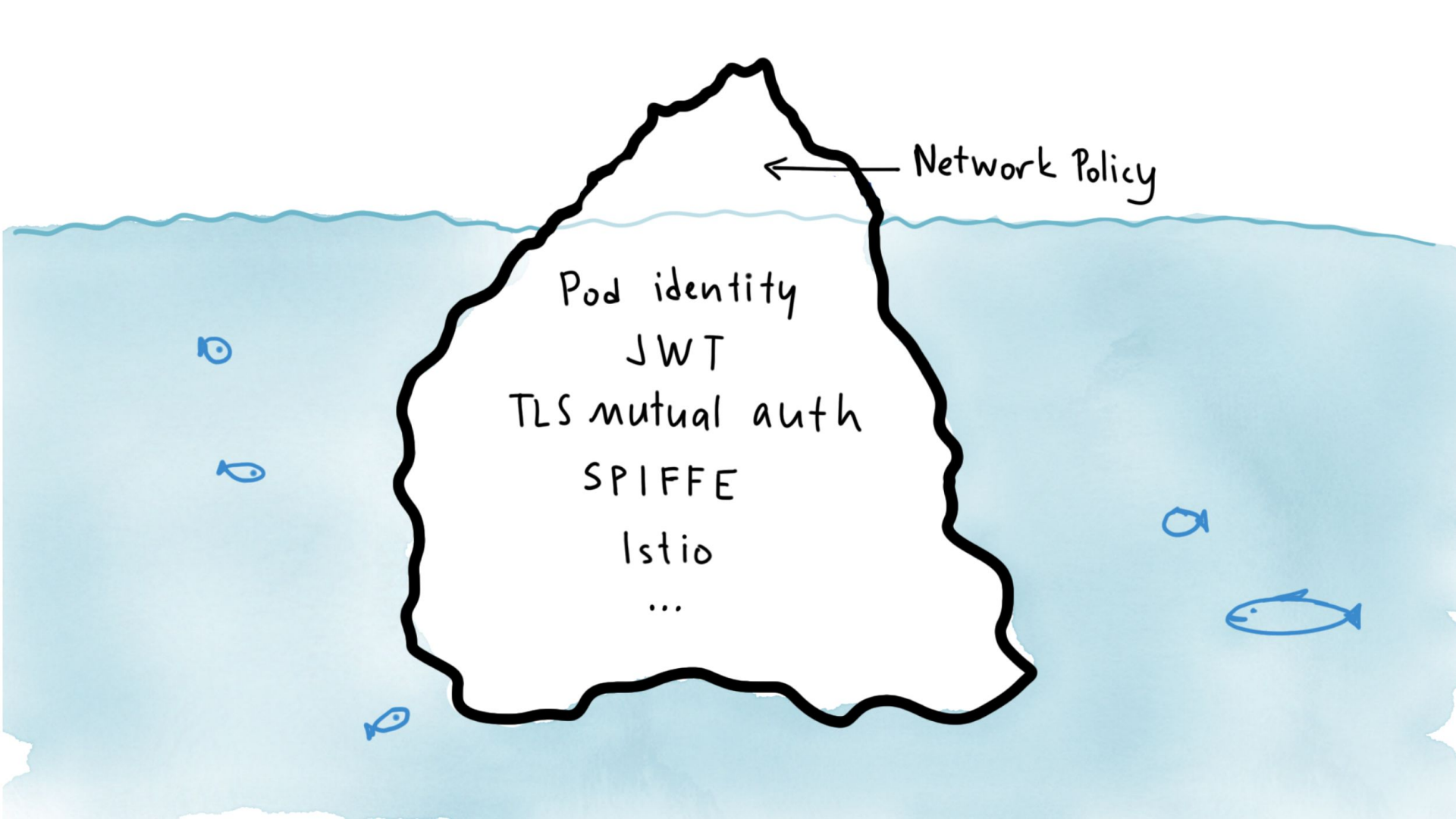
* try what's allowed/blocked
* test external connectivity
  - ingress: should I allow traffic from external LB?
  - egress: should I allow connections to external?
* test with other namespaces
  - allow connections from/to another namespace?
* use "kubectl describe" to verify rule syntax

learn more

github.com/ahmetb/kubernetes-networkpolicy-tutorial



recipes


Network Policy
Pod identity
JWT
TLS mutual auth
SPIFFE
Istio
...

thank you!

@ahmetb

software engineer @ Google Cloud

github.com/ahmetb/kubernetes-networkpolicy-tutorial

cloud.google.com/kubernetes-engine