

SECURING CLUSTERS WITH

Kubernetes Network Policies

@ahmetb

software engineer @ Google Cloud

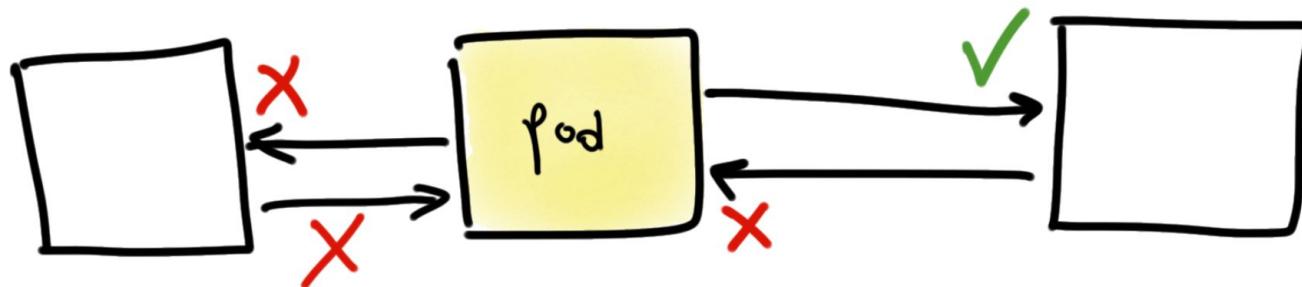


AGENDA

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



Network Policy API

pre-work → late 2015

alpha → v1.2 (March 2016)

beta → v1.3 (July 2016)

stable → v1.7 (June 2017)

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

writing
network
policies

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

Pod

app: shopping
tier: api

Pod

app: shopping
tier: db

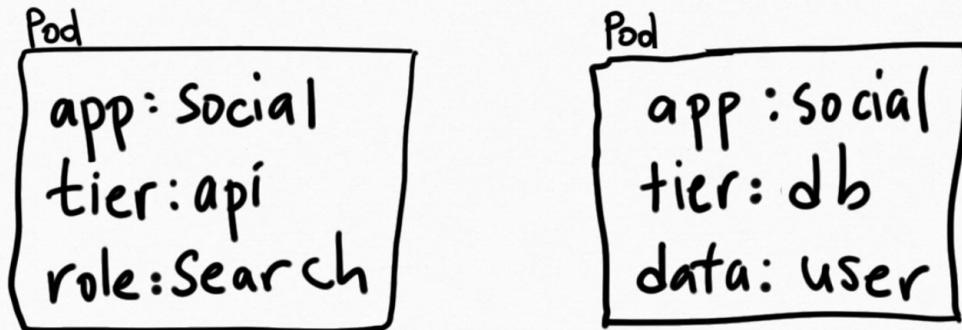
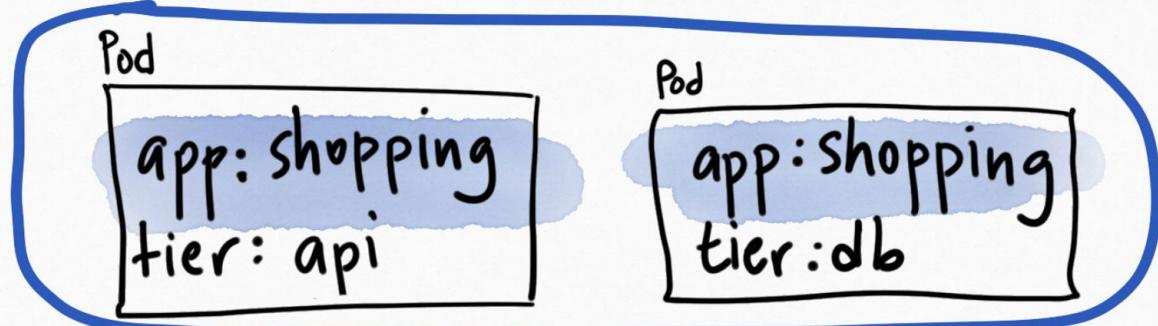
Pod

app: Social
tier: api
role: Search

Pod

app: social
tier: db
data: user

label selectors



matchLabels:

app: shopping

label selectors

Pod

app: shopping
tier: api

Pod

app: Social
tier: api
role: Search

Pod

app: shopping
tier: db

Pod

app: social
tier: db
data: user

matchLabels:

tier: db

empty label selector

Pod

app: shopping
tier: api

Pod

app: shopping
tier: db

matchLabels: {}

Pod

app: Social
tier: api
role: Search

Pod

app: social
tier: db
data: user

empty label selector

Pod

app: shopping
tier: api

Pod

app: shopping
tier: db

Pod

app: Social
tier: api
role: Search

Pod

app :social
tier: db
data: user

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?

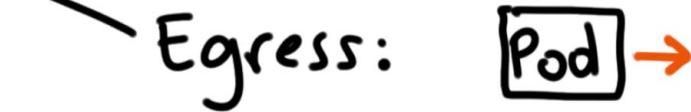
Ingress: 

Egress: 

anatomy of a Network Policy

① which pods does it apply to

② for which direction?



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.

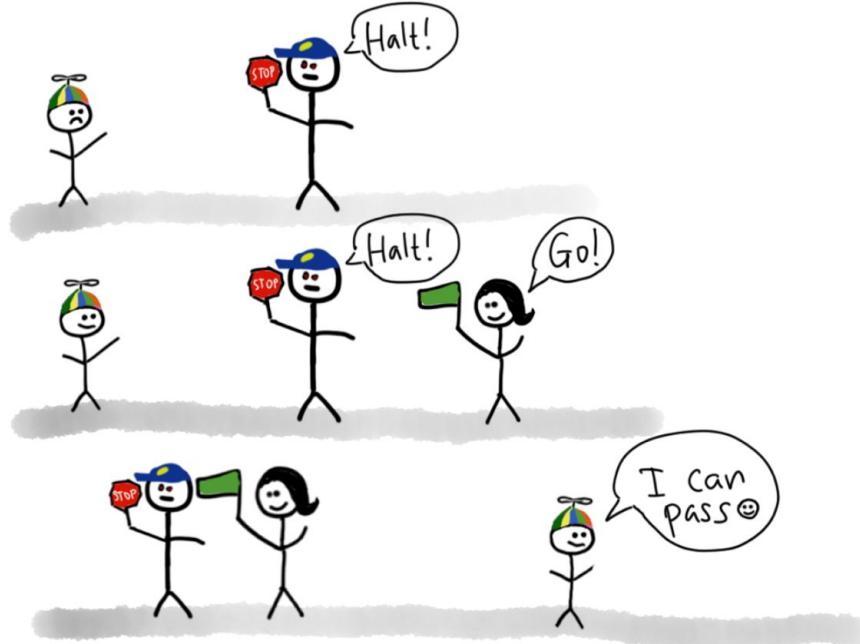
Denying all traffic by matching to it

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny-all
spec:
  podSelector: {} → select all pods
  ingress: [] → empty array indicates
                nothing is whitelisted
```

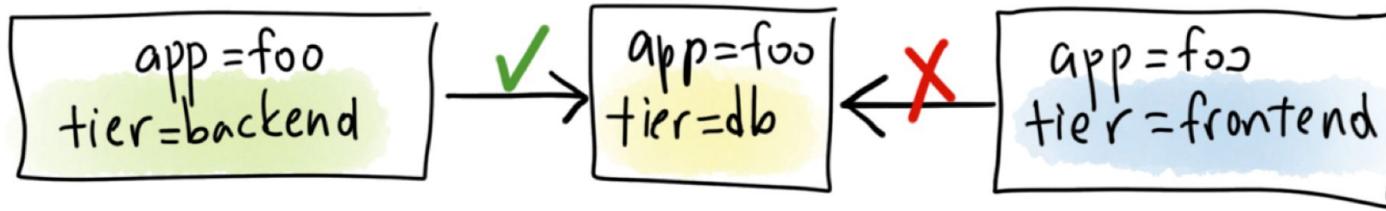
RESULT: all ingress traffic is blocked
by matching, and not allowing.

RULE ③

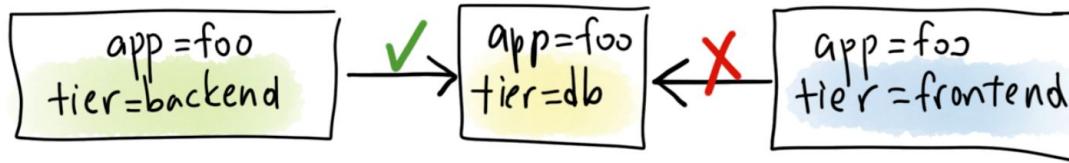
traffic is allowed if there is
at least one policy allowing it



Example : allowing some traffic



Example : allowing Some traffic



kind: NetworkPolicy

spec:

podSelector:

matchLabels:

app: foo

tier: db

ingress:

- from:

- podSelector:

matchLabels:

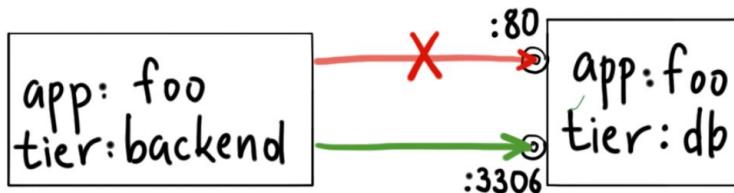
app: foo

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)

ingress:

-from: → rule #1

-podSelector:
matchLabels
app:foo
tier:backend

-podSelector:
matchLabels:
role:security

→ selector #1

→ selector #2

combined with OR
(not AND)

multiple rules are combined, too
(unioned)

ingress:

-from: → rule #1

-podSelector:
matchLabels
app:foo
tier:backend

-from: → rule #2

-podSelector:
matchLabels:
role:security



combined with OR operator
(not AND)

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."

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

Works only if the pods are in the same namespace!!!

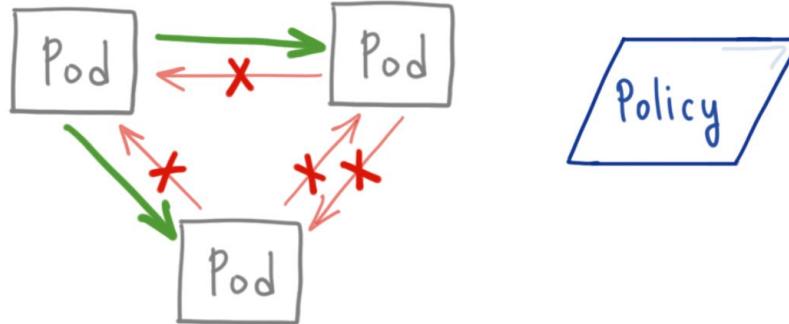
The diagram illustrates the configuration of an Ingress resource. It shows two sections: 'spec' and 'ingress'. In the 'spec' section, the 'podSelector' field is highlighted with a red oval. An arrow points from this oval to the 'role: monitoring' field in the 'matchLabels' section of an ingress rule. A red annotation in red text states: "Works only if the pods are in the same namespace!!!". A yellow oval highlights the 'port: 5000' field at the bottom of the ingress rule.

RULE ⑤

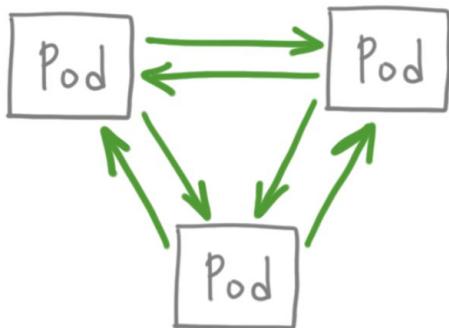
Network policies are scoped to
the namespace they're deployed to.

→ "spec.selector" 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.

namespace Selector

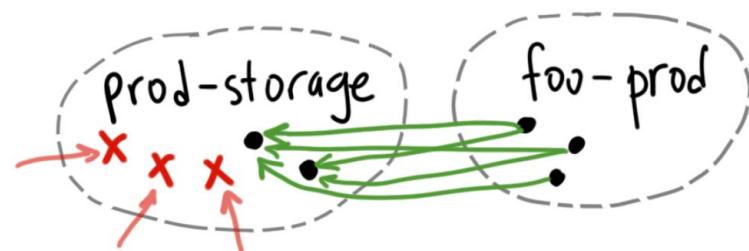
- * like podSelector
- * Selects namespaces, using labels

who is labeling their namespaces?

allowing traffic from other namespaces

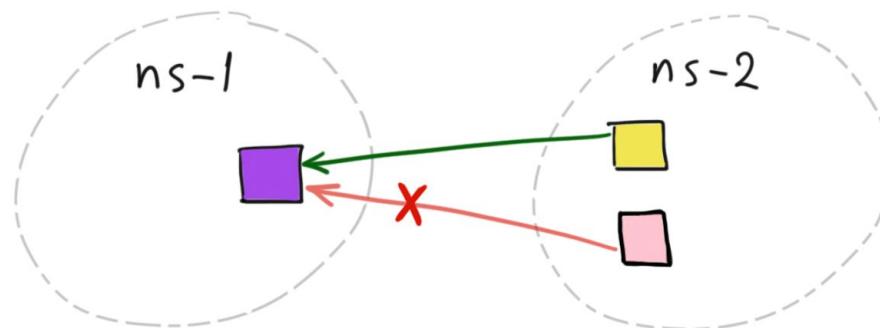
```
metadata:  
  namespace: prod-storage  
  name: allow-prod-apps  
  
spec:  
  podSelector: {}  
  
  ingress:  
    - from:  
      - namespaceSelector:  
          matchLabels:  
            purpose: prod
```

```
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 namespace Selector ✓

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

} from:
- ipBlock:
| | | cidr: 10.56.0.0/16

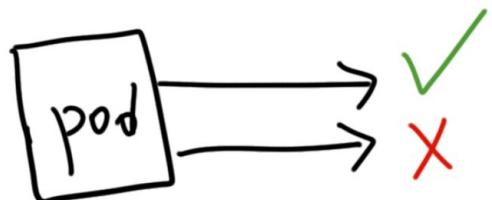
allow 10.x.x.x
but deny 10.11.12.x

} from:
- ipBlock:
| | | cidr: 10.0.0.0/8
| | except:
| | | 10.11.12.0/24

egress rules

almost identical to ingress rules

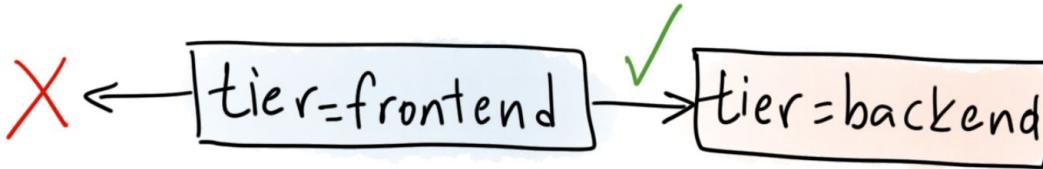
controls the traffic from selected pods



Denying all egress traffic

```
kind: NetworkPolicy
metadata:
  name: default-deny-all-egress
spec:
  podSelector: {} → select all pods in the namespace
  policyTypes: → need this if you have egress rules
    - Egress
  egress: [] → empty array = no rules.  
nothing is whitelisted
```

allowing some egress traffic



spec:

podSelector:

 matchLabels:

 tier: frontend

policyTypes:

- Egress

egress:

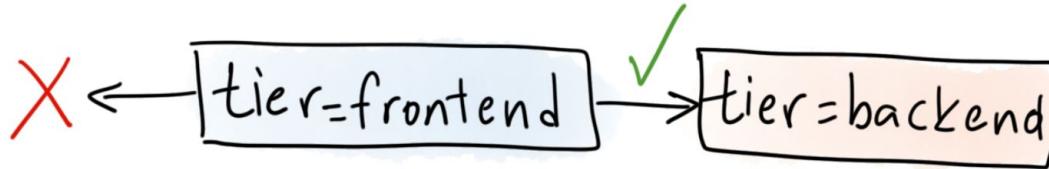
- to:

 - podSelector:

 matchLabels:

 tier: backend

allowing some egress traffic



spec:

```
podSelector:  
  matchLabels:  
    tier: frontend
```

```
policyTypes:  
- Egress
```

```
egress:  
- to:
```

```
- podSelector:  
  matchLabels:  
    tier: backend
```

(on frontend Pod)

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

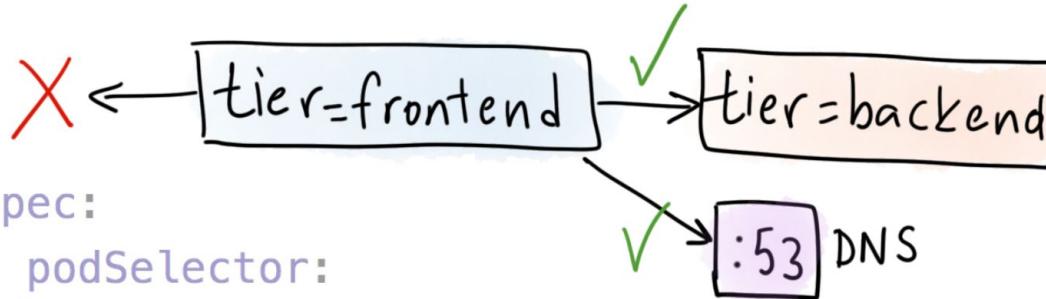
?????

egress policies actually
block dns resolution

you have to either

- allow all dns traffic (easier)
- add egress rule with
the IP address
of kube-dns (harder)

allowing some egress traffic & dns



spec:

 podSelector:

 matchLabels:

 tier: frontend

 policyTypes:

 - Egress

 egress:

 rule #1

 - to:

 - podSelector:

 matchLabels:

 tier: backend

 - ports:

 - port: 53

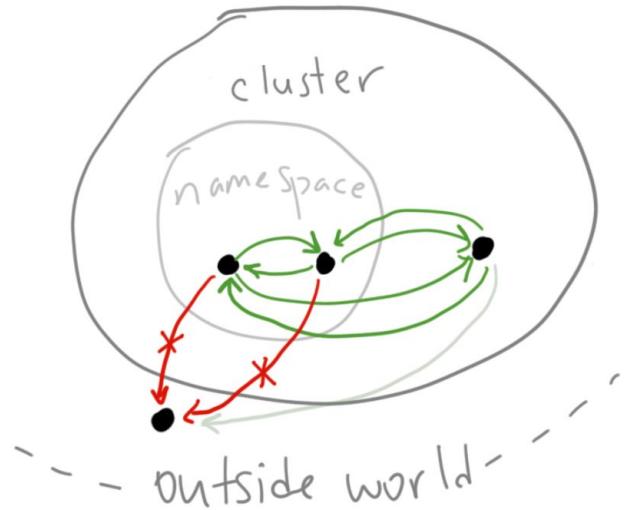
 | protocol: "UDP"

 - port: 53

 | protocol: "TCP"

 rule #2

more fun with egress blocking external traffic



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)



(beta)

GKE network policy

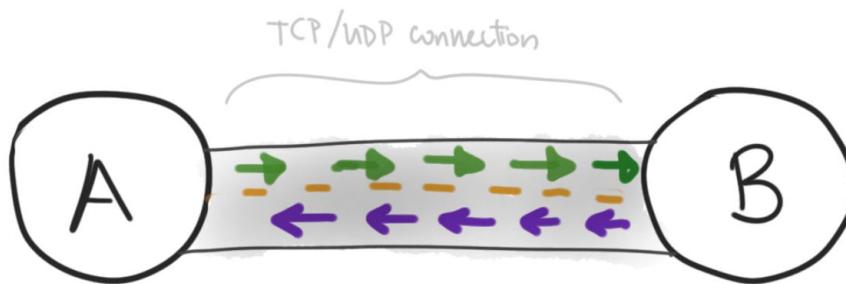
powered by Calico

```
$ 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)

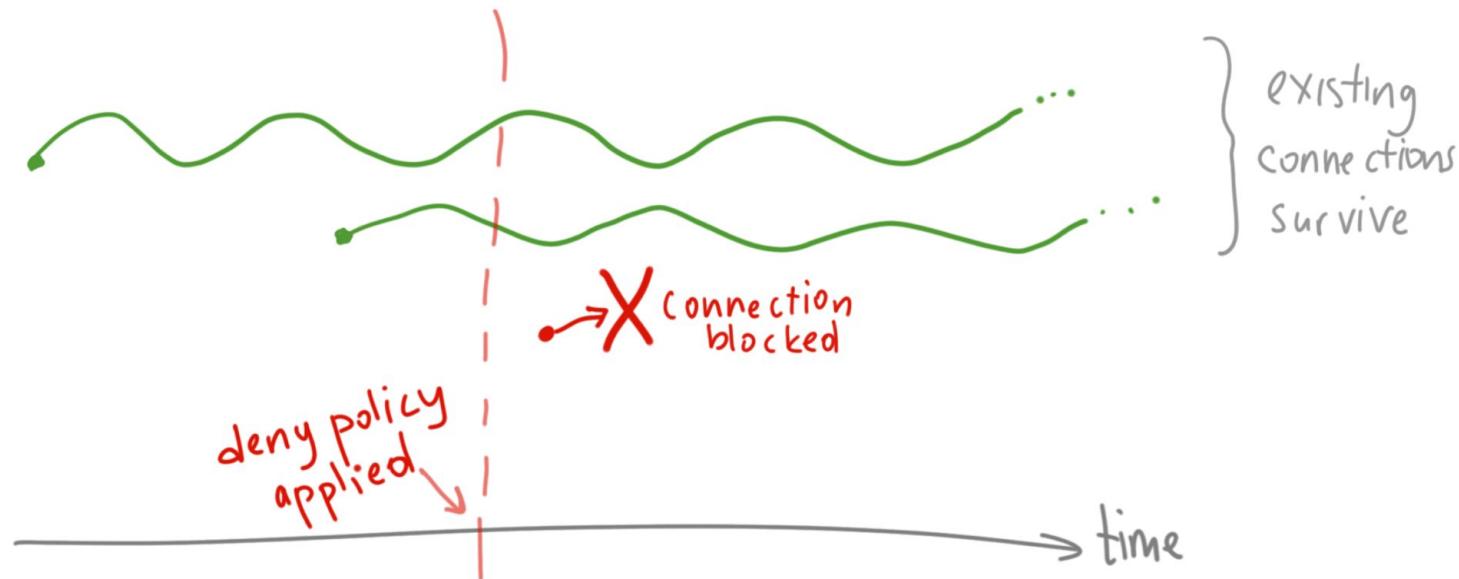
once A connects B, B can send data to A, on the same connection.



(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

```
ingress:  
- {}
```

vs

```
ingress:[]
```

vs

```
ingress:  
- from: {}
```

```
ingress:
```

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

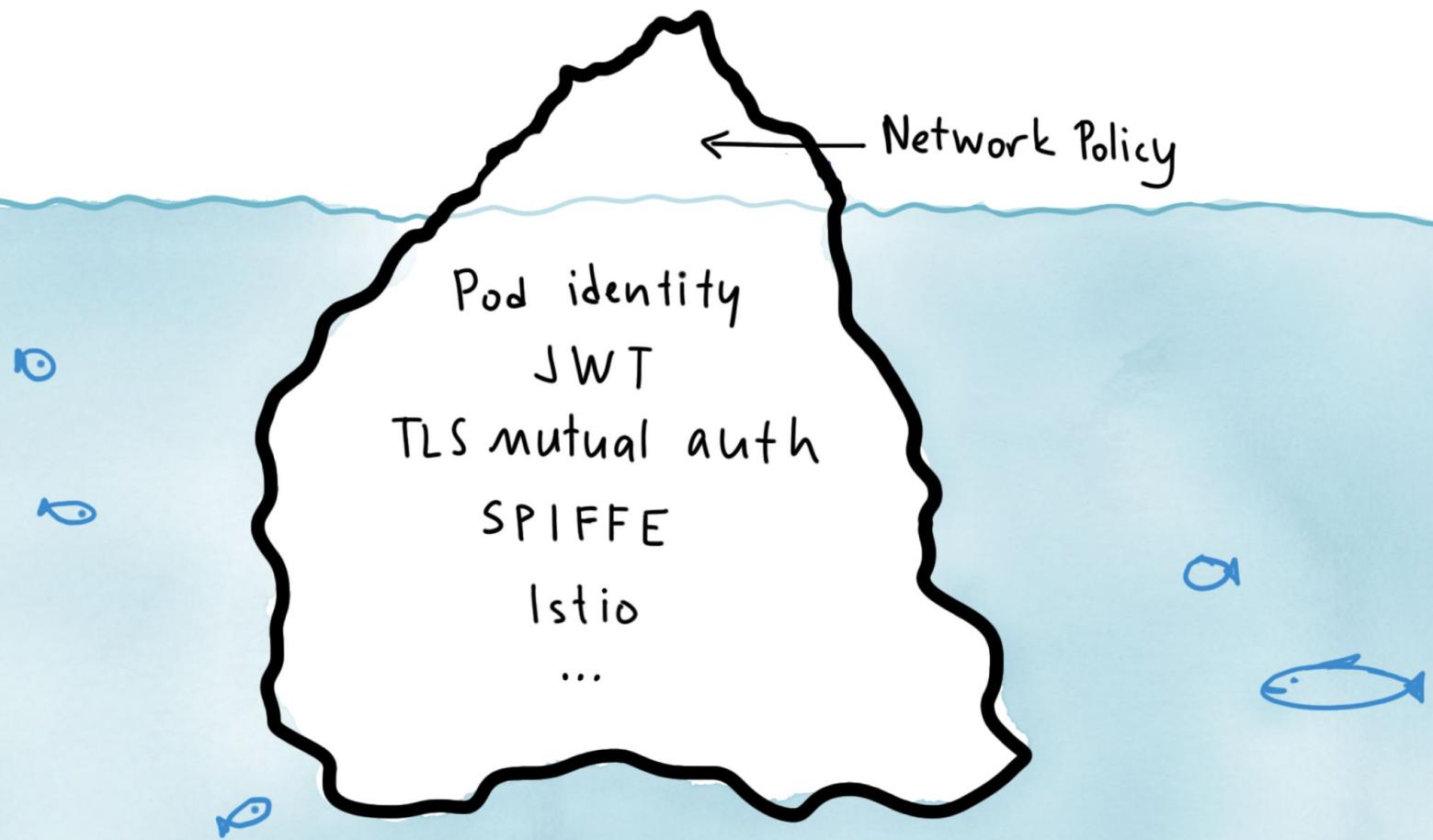
github.com/ahmetb/kubernetes-networkpolicy-tutorial

The screenshot shows the GitHub repository page for 'ahmetb / kubernetes-networkpolicy-tutorial'. The page has a dark header with a GitHub icon, 'This repository', 'Search', 'Pull requests', 'Issues', 'Marketplace', and 'Explore' buttons. On the right, there are notification, add, and user icons. Below the header, the repository name is displayed with a fork icon. To the right are buttons for 'Unwatch' (with 10 notifications), 'Unstar' (with 124 stars), and 'Fork' (with 17 forks). A horizontal line separates this from the main content area.

Recipes for securing your cluster with Kubernetes Network Policies

⌚ 79 commits	⌚ 1 branch	⌚ 0 releases	👤 1 contributor	Apache-2.0
ahmetb Some egress rules ...	Latest commit a18f9e6 23 hours ago			
📄 img	Update 4.gif			4 months ago
📄 00-create-cluster.md	Add 02a-allow-all-traffic-to-an-application			2 days ago
📄 01-deny-all-traffic-to-an-application.md	Add 02a-allow-all-traffic-to-an-application			2 days ago
📄 02-limit-traffic-to-an-application.md	Add 02a-allow-all-traffic-to-an-application			2 days ago
📄 02a-allow-all-traffic-to-an-application.md	Add 02a-allow-all-traffic-to-an-application			2 days ago
📄 03-deny-all-non-whitelisted-traffic-in-the-namespace.md	Add 02a-allow-all-traffic-to-an-application			2 days ago
📄 04-deny-traffic-from-other-namespaces.md	Small fixes			2 days ago
📄 05-allow-traffic-from-all-namespaces.md	Add 02a-allow-all-traffic-to-an-application			2 days ago
📄 06-allow-traffic-from-a-single-namespace.md	Add 02a-allow-all-traffic-to-an-application			4 months ago

recipes



thank you!

@ahmetb

software engineer @ **Google** Cloud



github.com/ahmetb/kubernetes-networkpolicy-tutorial

cloud.google.com/kubernetes-engine