
KubeCon
CloudNativeCon
North America 2023


KubeCon | CloudNativeCon

North America 2023

# Network Policy API:

## Intro and Project Update

*Andrew Stoycos, Dan Winship, Surya Seetharaman, Red Hat*

*Yang Ding, VMware*




# Overview



- What is the network-policy-api subgroup? - Andrew
  - How did we get here
    - (link to last year's contributor summit talk so we can pull stuff from that…)
- NetworkPolicy API - Dan
- AdminNetworkPolicy API
  - A brief history (KEP), later work - Yang
  - Status - Yang
  - Implementations - Antrea, OVNK
    - (encourage more implementations)
- Future work
  - NPEPs, ANP v1beta1, etc - Surya (will start the format idea and delegate)
  - conformance tests + collaboration with GWAPI + conformance profiles- Surya
  - "NPv2" / DNP -Dan
- How and why you should get involved Whether you are a K8s user, developer, administrator, operator or CNI maintainer


KubeCon | CloudNativeCon

North America 2023

# sig-network-policy-api subgroup

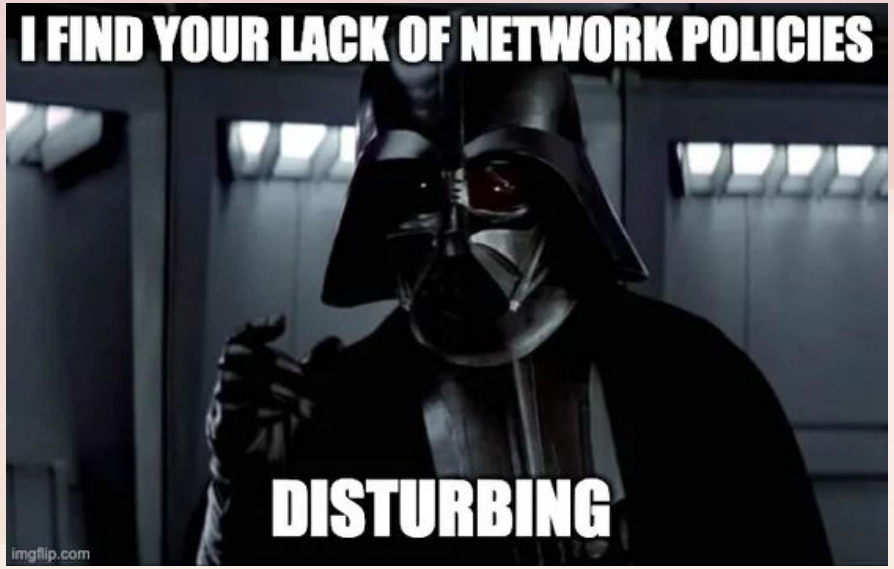
# sig-network-policy-api: Who are we?



- **Who are we?**
  - A passionate group of developers who spawned out of the Kubernetes Networking Special interest group, focused explicitly on the maintenance and development of policy related APIs
  - https://github.com/kubernetes-sigs/network-policy-api/graphs/contributors + many more :)

- **What do we do?**
  - Help maintain the legacy NetworkPolicy v1 resource
  - Rework of the upstream network policy test suite
  - The AdminNetworkPolicy KEP
  - Stay tuned…

# NetworkPolicy API



# NetworkPolicy v1: Designed for Developers



As an **application developer**, I want to control which other users within the cluster can access my application

As an **application developer**, I want to have a multi-tiered architecture, where I can control access between pods in each tier

# NetworkPolicy v1: Designed for Developers




In 2016 we were all still drunk on DevOps, and Kubernetes mostly didn't even have clear "administrator" / "user" distinctions.

(RBAC was added in the same Kubernetes release as NetworkPolicy.)

# NetworkPolicy v1



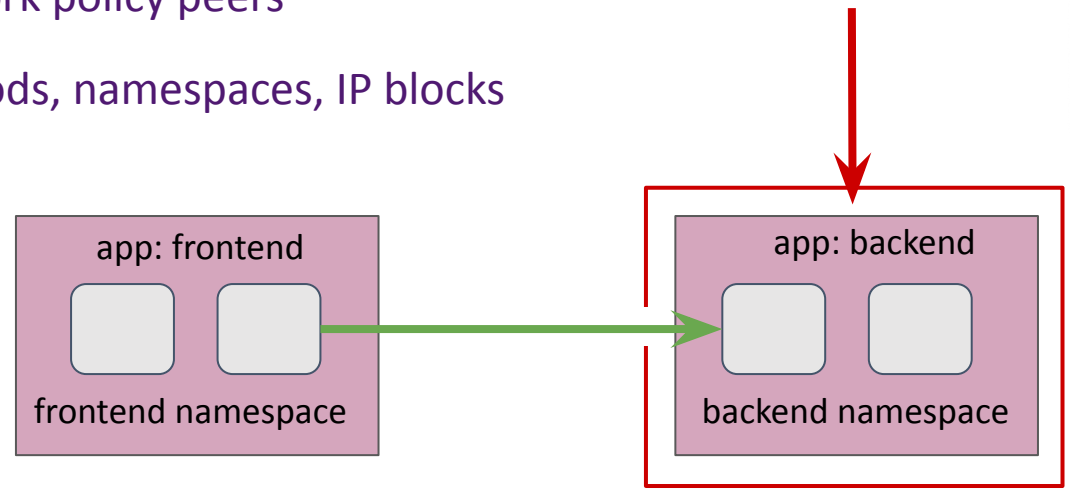
- How can app owners control traffic to/from their workloads?
  - example: backends can get traffic only from frontends, databases can only get traffic from backends etc..
- Namespace scoped Policy API that lets users define simple ingress/egress rules
- API design is implicit in nature
- Network policy peers
  - pods, namespaces, IP blocks

np.yaml — ~/Documents

np.yaml

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-ingress-to-backend-from-frontend
  namespace: backend-1
spec:
  podSelector:
    matchLabels:
      app: backend
  policyTypes:
  - Ingress
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          app: frontend

```



# NetworkPolicy v1



- Has been a stable v1 API for over 6 years
- Numerous implementations exist, below are just a few







# NetworkPolicy v1: Issues

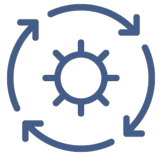


Policies only specify what is *allowed* and leave everything else implicitly denied in a way that is confusing and hard to work with

**Implicit isolation model**

The lack of deny rules, priority, and namespace-spanning policies, means it doesn't solve administrator use cases

**Not for admin use cases**



Behavior of policy as it applies to traffic entering/exiting a cluster is underspecified

**N/S traffic selection**

# Time for a new API!

## The KEP process

- 10+ contributors
- 100+ individual commits
- 600+ comments resolved
- 1 year from PR open to merge

# AdminNetworkPolicy API - KEP



## Contributors / reviewers https://github.com/kubernetes/enhancements/pull/2522


Casey


Gobind


Ricardo


Abhishek


Dan

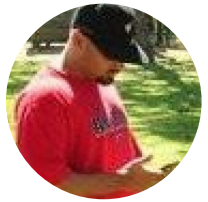

Satish


Tim


Sanjeev


Thomas


Andrew


Jay


Yang


Vinay


Christopher

# AdminNetworkPolicy API - KEP



## Motivation

A cluster-scoped, admin-facing, CNI-agnostic policy API for complete control of cluster networking

## Challenges

- Consolidate on use cases (which were a lot)
- Need to be explicit in API design
- Priority modeling
- Provide rule actions that make sense to everyone
- Iterate design so that the API is minimal but solves all use cases agreed on
- Future extensibility

# AdminNetworkPolicy API

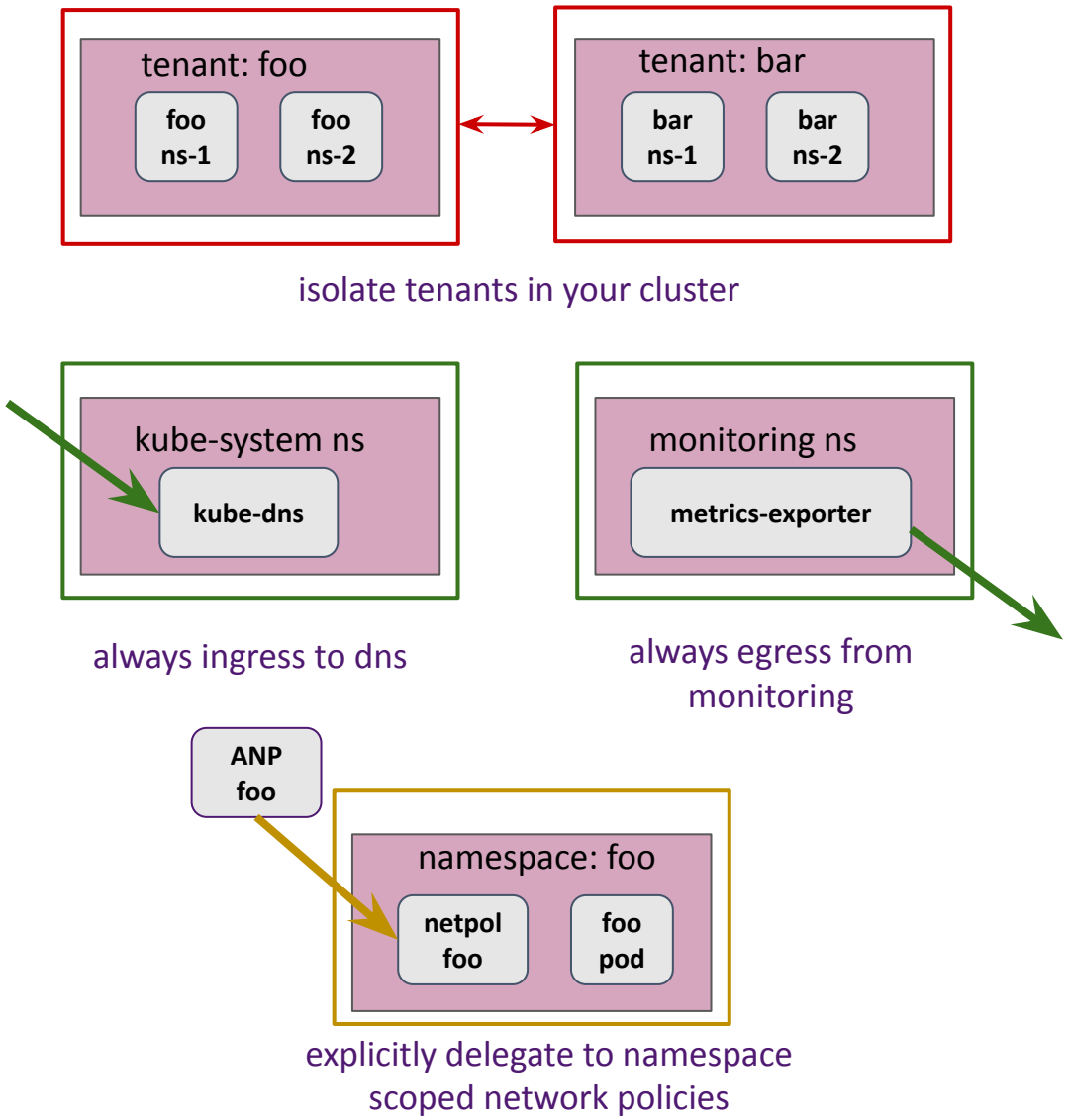
## Started from the KEP now we're here

- Two objects: AdminNetworkPolicy and BaselineAdminNetworkPolicy
- out-of-tree
- v1alpha
- Supports intra-cluster controls

# AdminNetworkPolicy API



- How can cluster admins control traffic to/from their clusters' workloads in a non-overridable manner?
  - isolate tenants
  - always allow ingress to DNS namespace
  - always allow egress from monitoring namespace
  - delegate to namespace scoped network policies
  - always deny traffic to sensitive namespace from everywhere



# AdminNetworkPolicy API



- Let admins define simple ingress/egress rules to secure their cluster that takes precedence over NetworkPolicy API
- There's also BaselineAdminNetworkPolicy to define the default security posture of a cluster in absence of NetworkPolicies
- API design is explicit in nature

np.yaml — ~/Documents

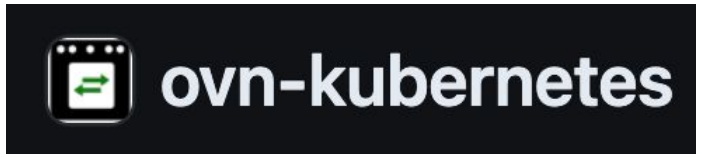
anp.yaml

```yaml
apiVersion: policy.networking.k8s.io/v1alpha1
kind: AdminNetworkPolicy
metadata:
  name: cluster-wide-allow-example
spec:
  priority: 30
  subject:
    namespaces: {}
  ingress:
    - action: Allow
      from:
      - namespaces:
          namespaceSelector:
            matchLabels:
              kubernetes.io/metadata.name: monitoring-ns
  egress:
    - action: Allow
      to:
      - pods:
          namespaces:
            namespaceSelector:
              matchlabels:
                kubernetes.io/metadata.name: kube-system
          podSelector:
            matchlabels:
              app: kube-dns
```

# AdminNetworkPolicy Current Status



- Has been a v1alpha1 API for over 1 year; on the journey towards beta….
- Two implementations exist, we welcome more implementations and call for feedback!!


- On the roadmap of:




1. Calico tracker: https://github.com/projectcalico/calico/issues/7578
2. Cilium tracker: https://github.com/cilium/cilium/issues/23380
3. KubeOVN Tracker: https://github.com/kubeovn/kube-ovn/issues/3247

# Network Policy Enhancement Proposals + Feature development

- Similar to KEPs but for the sig-network-policy-api project…
- Have a cool idea? Open a NPEP!

# NPEP for Conformance Profiles



- How can API maintainers track implementations and establish a feedback loop?
- How can implementations be more involved in the API conformance design?
- NPEP proposes to add a new **ConformanceReport** CRD that captures the results of each *core* and *extended* feature **profile**

```
profiles:
- core:
    failedTests:
    - AdminNetworkPolicyIngressSCTP
    - AdminNetworkPolicyEgressUDP
    result: failure
    statistics:
      Failed: 2
      Passed: 5
      Skipped: 0
      summary: ""
    name: AdminNetworkPolicy
- core:
    result: success
    statistics:
    Failed: 0
    Passed: 7
    Skipped: 0
    summary: ""
    name: BaselineAdminNetworkPolicy
```

NPEP 137

Reports Folder

**Help Wanted!!**

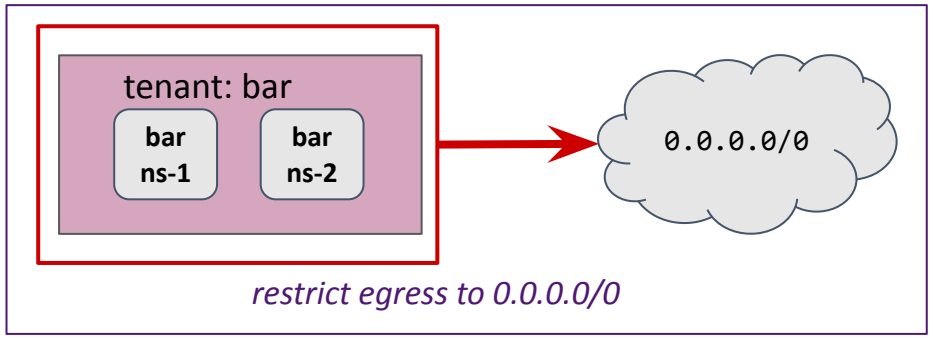
Contributor talk

Surya
@tssurya

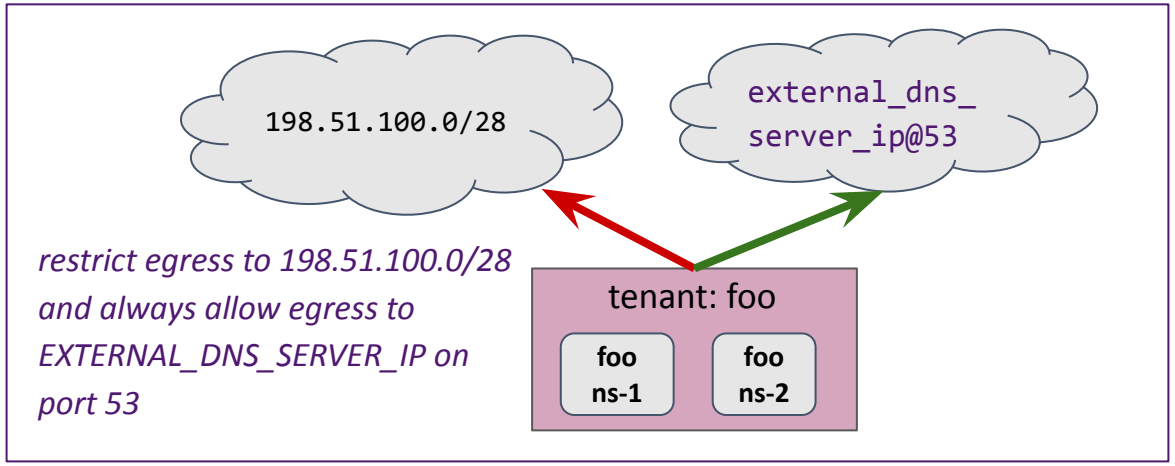
# NPEP for cluster egress traffic controls

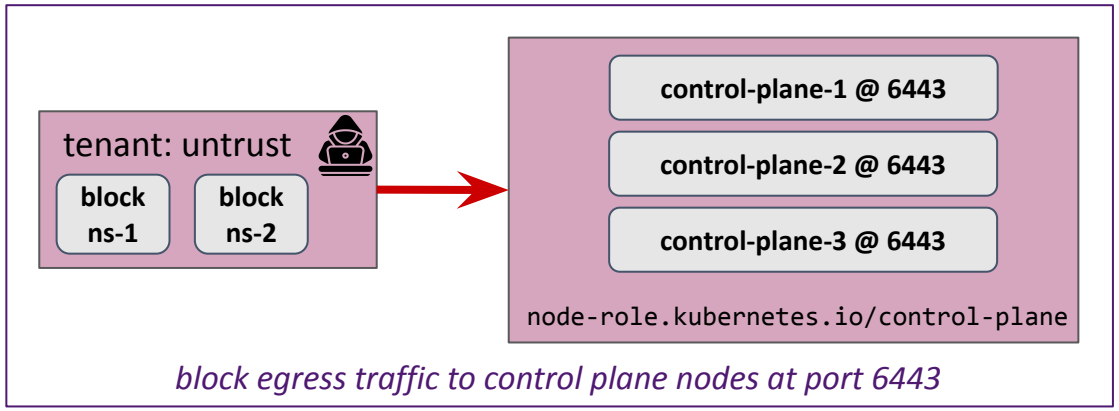


- Current API has two egress peers: **namespaces** and **pods**
- How can cluster admins control northbound traffic from their clusters' workloads?




NPEP 126




- NPEP proposes to add two new egress peers - **nodes** and **networkCIDRs**

**Help Wanted!!**
**Leave feedback!**
**Comment on the Issue!**

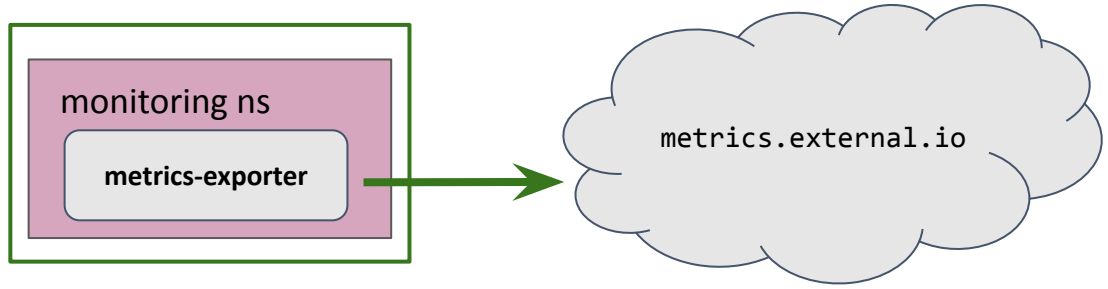
Surya
@tssurya

# NPEP for cluster egress FQDN traffic controls

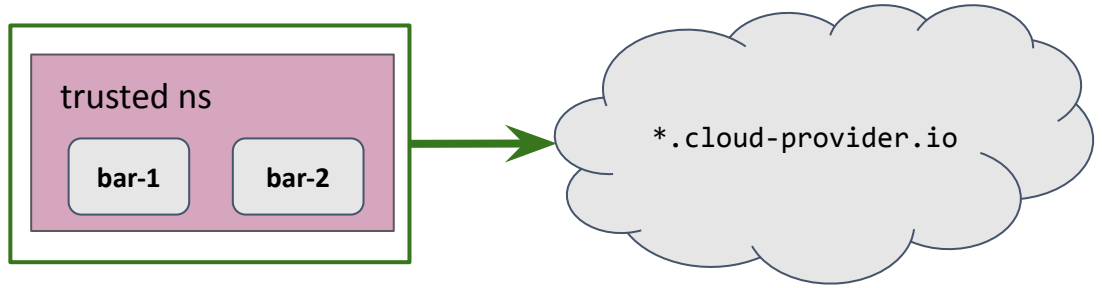


- Proposal: Allow specifying Fully-Qualified Domain Names (for example www.kubernetes.io) as egress peers
  - Also allows for wildcard selectors
- Caveats:
  - Only focused on egress
  - Only “ALLOW” type rules are permitted. The NPEP proposes prohibiting “DENY” type rules with a FQDN peer.
  - Not a true L7 policy, but rather uses DNS information for L3/4 enforcement

NPEP 133




allow metrics-exporter to communicate with an external metrics aggregator




allow trusted namespace to call all APIs provided by a cloud provider

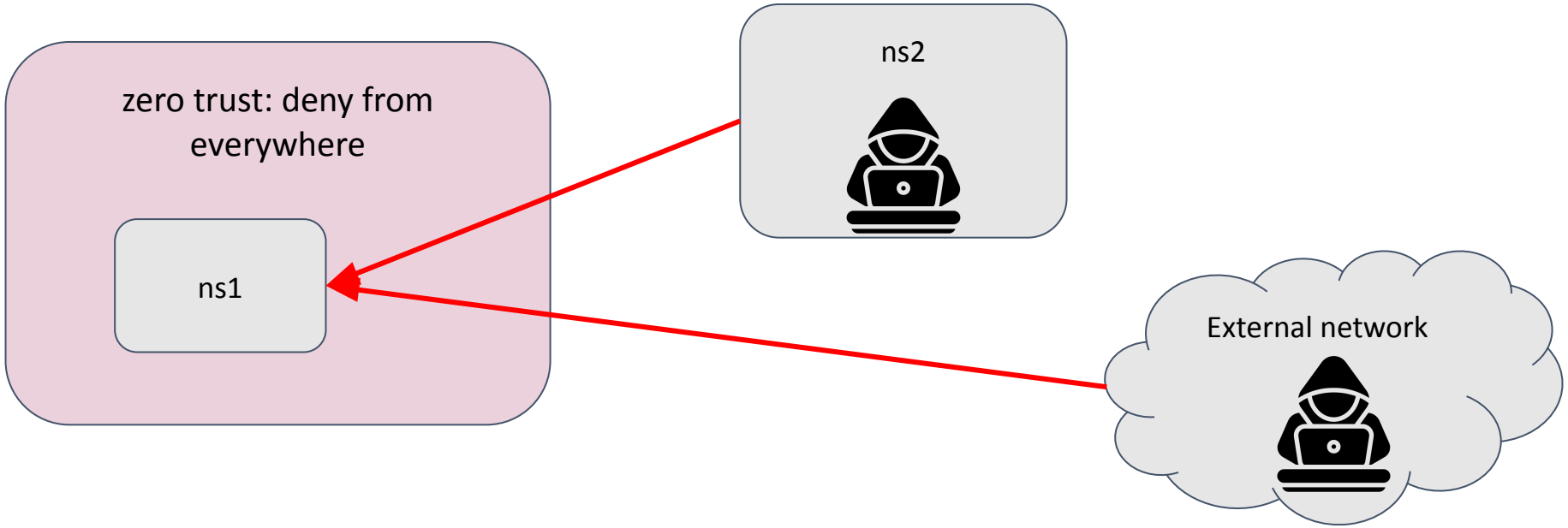
Rahul
@rahulkjoshi

# NPEP for cluster Ingress traffic controls



- Problem: Existing API doesn't allow to specify external endpoints for ingress.
- Goal: Implement ingress traffic control from external destinations (outside the cluster)
- Issue: https://github.com/kubernetes-sigs/network-policy-api/issues/127

NPEP 128



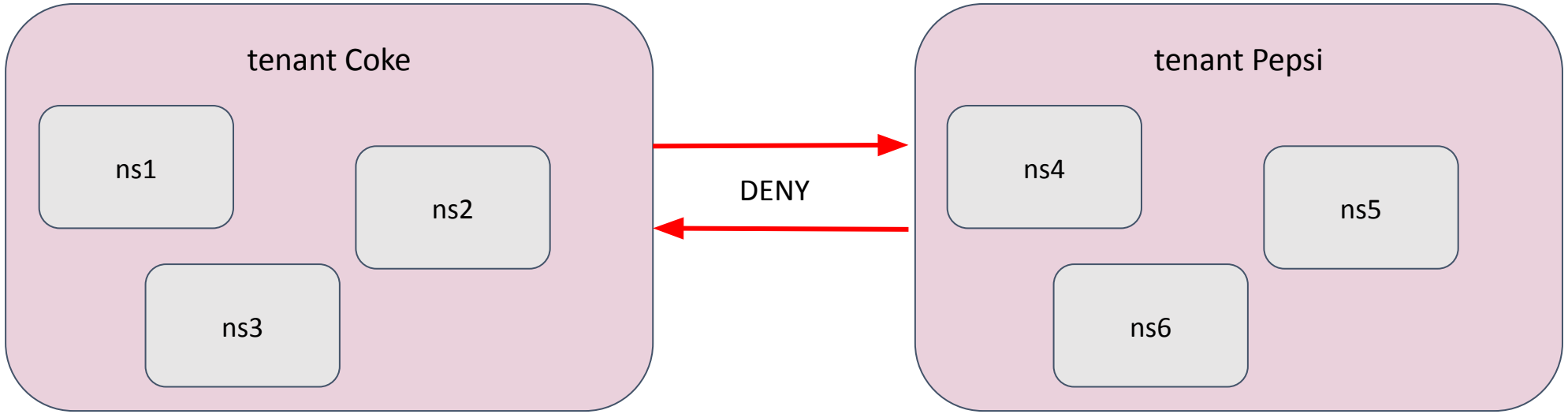

Nadia
@npinaeva

# NPEP for tenancy



- Problem: Existing API allows much more configurations that were initially required, which may be confusing for users.
- Goal: Simplify existing API to only allow configurations that are required by the outlined user stories.
- Issue: https://github.com/kubernetes-sigs/network-policy-api/issues/122

NPEP 122




Nadia
@npinaeva

# Policy Assistant Feature (#150) (built on Cyclonus)



## *Goal 1: Develop/Understand your Policies*

- Predict verdict from policies:

| TRAFFIC | VERDICT | INGRESS FLOW | EGRESS FLOW |
|---|---|---|---|
| test/client → prod/server:80 (TCP) | Denied by ANP | [ANP] default/anp1: deny (pri=3) | [NPv1] prod/np1: deny |
| prod/client → prod/server:81 (UDP) | Allowed by BNP | [ANP] default/anp1: pass (pri=44)<br>[BNP] default/bnp: allow (pri=55) | (No matching rules) |

- Explain policies:

| TYPE | TARGET | SOURCE RULES | PEER | ACTION | PORTS |
|---|---|---|---|---|---|
| Ingress | namespaces with same labels for “key1” | [ANP] default/anp1<br>[BNP] default/bnp | all namespaces<br>pods w/ label “role=client” | ANP:<br>allow (pri=3),<br>deny (pri=4),<br>pass (pri=7)<br><br>BNP:<br>allow (pri=1) | 80-99, TCP |
| | | | namespaces w/ label “user=3”<br>all pods | ANP:<br>deny (pri=9) | |
| | namespace: default<br>pods w/ label “a=b” | [NPv1] default/np1<br>[NPv1] default/np2 | namespace: default<br>pods w/ label “role=client” | NPv1:<br>peers allowed | 80, UDP |

## *Goal 2: Extended Test Framework*

- Run hundreds of test scenarios, filtering by tag (e.g. run ANP tests).
- Quick-to-code tests automatically calculate expected connectivity from policies.

Hunter
@huntergregory


# NetworkPolicy v2?

# NetworkPolicy v2?



- AdminNetworkPolicy solves the fact that NP doesn't implement administrator use cases, but it still leaves NP's other problems:
  - "Implicit deny on first policy in the namespace" is hard to work with.
  - Weird syntactic quirks make NetworkPolicies hard to write and understand.
  - Backward compatibility requires us to leave a hole in policies for kubelet probes.
  - It's hard for SIG Network to add new features to NetworkPolicy without causing old implementations to misinterpret new NetworkPolicy objects.
- A literal `networking.k8s.io/v2` version of NetworkPolicy wouldn't help, because of backward-compatibility constraints.

- ANP is "better NetworkPolicy for administrators". Could we also create "better NetworkPolicy for developers"?

# DeveloperNetworkPolicy

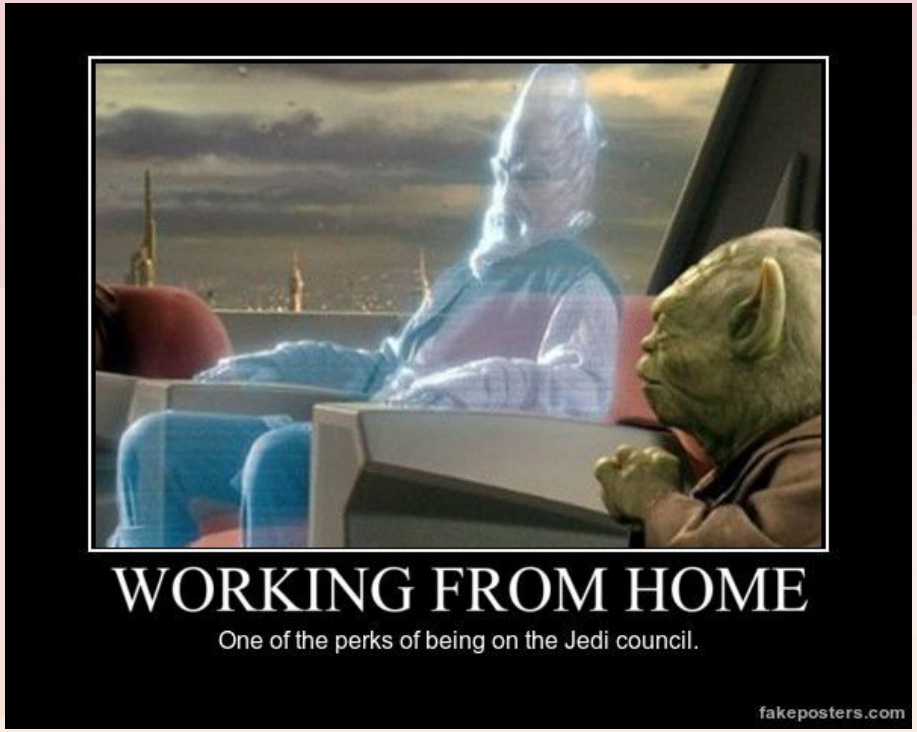


- (Still just an idea: NPEP-136.)

NPEP 136

- Let's take a step back from what NetworkPolicy became, and reinvent developer-focused networking policy from the user stories up.
  - Don't let administrator-focused use cases sneak in, as happened with NetworkPolicy; keep it solely focused on developer use cases.
  - Ensure that it fits perfectly between AdminNetworkPolicy and BaselineAdminNetworkPolicy, allowing use cases that are harder to implement with NetworkPolicy.
- AdminNetworkPolicy + DeveloperNetworkPolicy = "NetworkPolicy v2"
  - NetworkPolicy v1 would still keep working (at least, as long as implementors wanted to support it), but it would stop getting new features, and we would push people toward ANP and DNP.

# How To Get Involved



# Getting Involved in the project!

KubeCon CloudNativeCon North America 2023

- **Level 1: Completely new to K8s and Don't know where to start!?**
  - Checkout the K8s contributor guide!
  - Read the Network Policy V1 documentation
  - Read the AdminNetworkPolicy KEP
- **Level 2: Familiar with the basics but still unsure where to dive in?**
  - Read though our documentation at https://network-policy-api.sigs.k8s.io/ please open issues I am sure there are documentation bugs we've yet to find 😈
  - Checkout our help wanted and good first issues
  - Review any of the open PRs
- **Level 3: Want more features! Want more implementations!**
  - Read about our NPEP (Network Policy Enhancement Proposal) process
  - Propose a new NPEP with an issue.

kubernetes-sigs/network-policy-api

#sig-network-policy-api

https://network-policy-api.sigs.k8s.io

**Please scan the QR Code above**
**to leave feedback on this session**