Kubernetes Advanced

kubernetes

Session #05
Network Policies

kubernetes

Session Contents

- Network Policies
- Pod Isolation
- Sample Network Policies
- Network Plugins
- Best Practices

kubernetes

Network Policies

Motivation

- By default, your pods can reach and be reached by any other pod running inside your cluster
- Usually you have your cluster shared by different environment and/or different solutions
- Is important to have a fine-grained way to define egress and ingress policies inside your cluster
- NetworkPolicies allow you to do that

Network Policies

- NetworkPolicies allow you to specify how a pod is allowed to communicate with various network "entities" (services, pods) over the network
- The entities that a Pod can communicate with are identified through a combination of the following 3 identifiers, stating allowed connections:
    - Other pods (exception: a pod cannot block access to itself)
    - Namespaces
    - IP blocks (exception: traffic to and from the node where a Pod is running is always allowed, regardless of the IP address of the Pod or the node)

kubernetes

Network Policies

- When defining a pod- or namespace- based NetworkPolicy, you use a selector to specify what traffic is allowed to (ingress) and from (egress) the Pod(s) that match the selector
- Meanwhile, when IP based NetworkPolicies are created, we define policies based on IP blocks (CIDR ranges)
- However, like ingress, Kubernetes don’t have a native implementation of NetworkPolicies
- Network policies are implemented by the network plugin which supports NetworkPolicy
- Creating a NetworkPolicy resource without a controller that implements it will have no effect.

kubernetes

Pod Isolation

Pod Isolation

- There are two sorts of isolation for a pod: isolation for egress, and isolation for ingress
- The two sorts of isolation (or not) are declared independently, and are both relevant for a connection from one pod to another
- By default, a pod is non-isolated for egress; all outbound connections are allowed
- By default, a pod is non-isolated for ingress; all inbound connections are allowed

kubernetes

Pod Isolation: Egress

- A pod is isolated for egress if there is any NetworkPolicy that both selects the pod and has "Egress" in its policyTypes
- When a pod is isolated for egress, the only allowed connections from the pod are those allowed by the egress list of some NetworkPolicy that applies to the pod for egress
- Reply traffic for those allowed connections will also be implicitly allowed
- The effects of those egress lists combine additively.

Pod Isolation: Ingress

- A pod is isolated for ingress if there is any NetworkPolicy that both selects the pod and has "Ingress" in its policyTypes
- When a pod is isolated for ingress, the only allowed connections into the pod are those from the pod's node and those allowed by the ingress list of some NetworkPolicy that applies to the pod for ingress
- Reply traffic for those allowed connections will also be implicitly allowed
- The effects of those ingress lists combine additively.

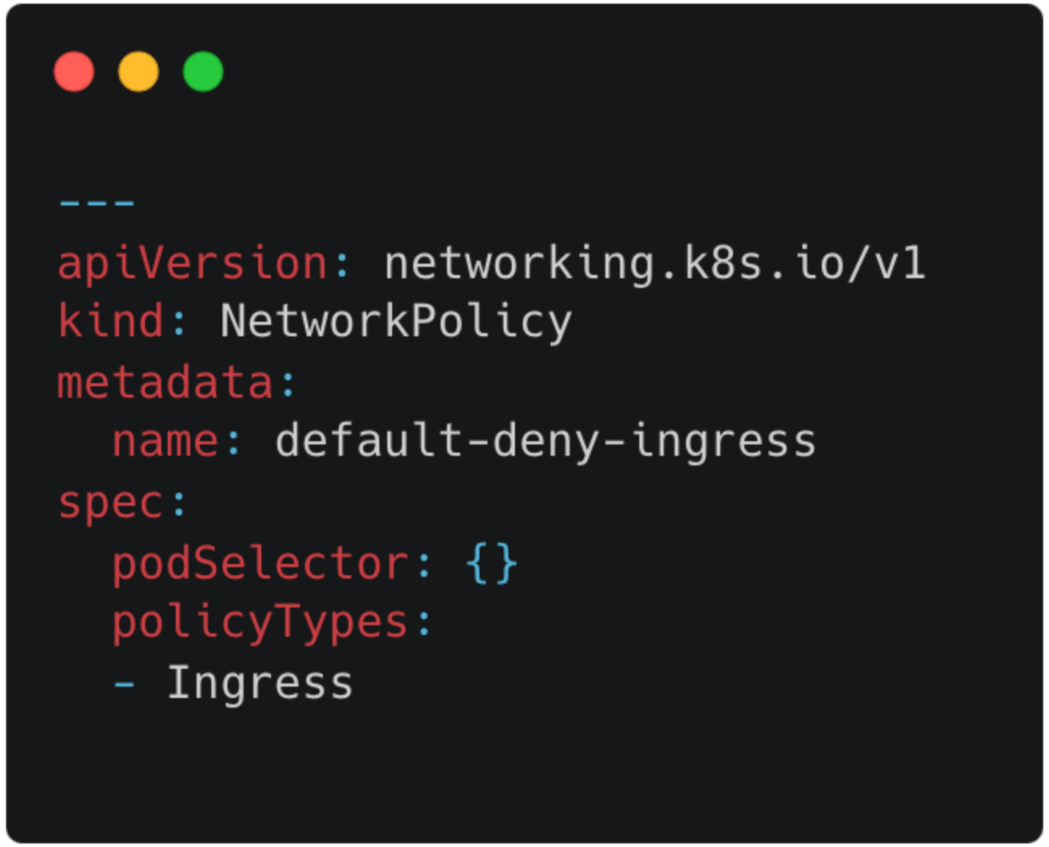
Pod Isolation: Several Policies

- Network policies do not conflict; they are additive
- If any policy or policies apply to a given pod for a given direction, the connections allowed in that direction from that pod is the union of what the applicable policies allow
- Thus, order of evaluation does not affect the policy result.
- For a connection from a source pod to a destination pod to be allowed, both the egress policy on the source pod and the ingress policy on the destination pod need to allow the connection. If either side does not allow the connection, it will not happen.

kubernetes

Samples

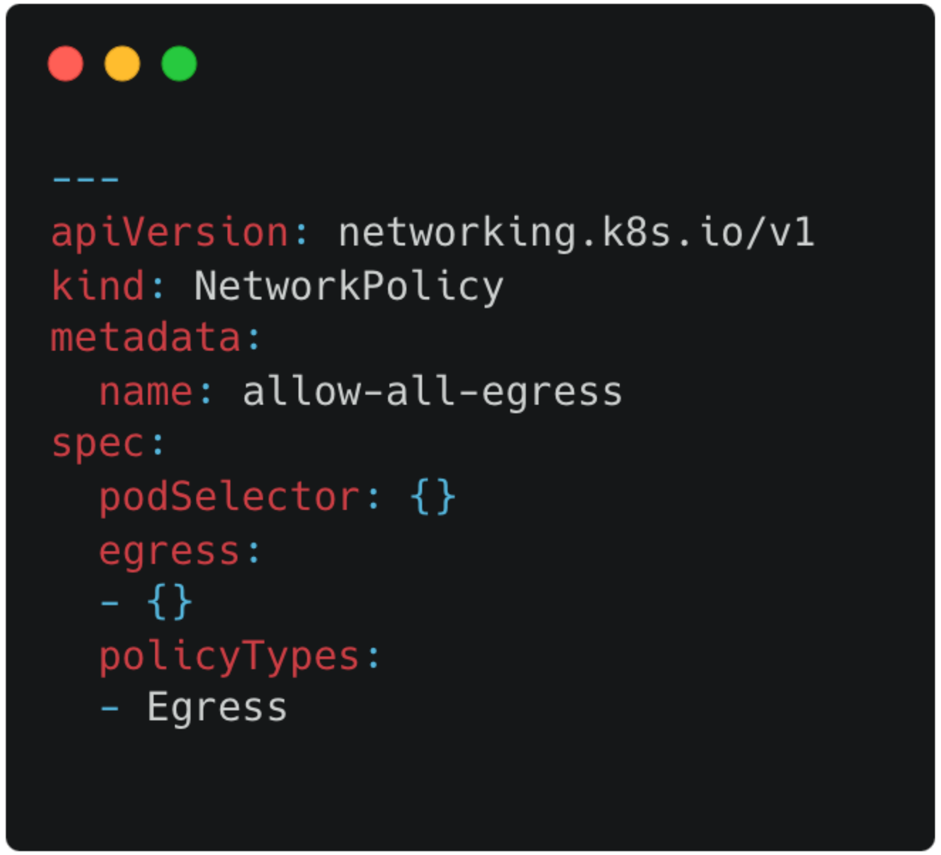
Policy: Default deny all ingress traffic



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

kubernetes

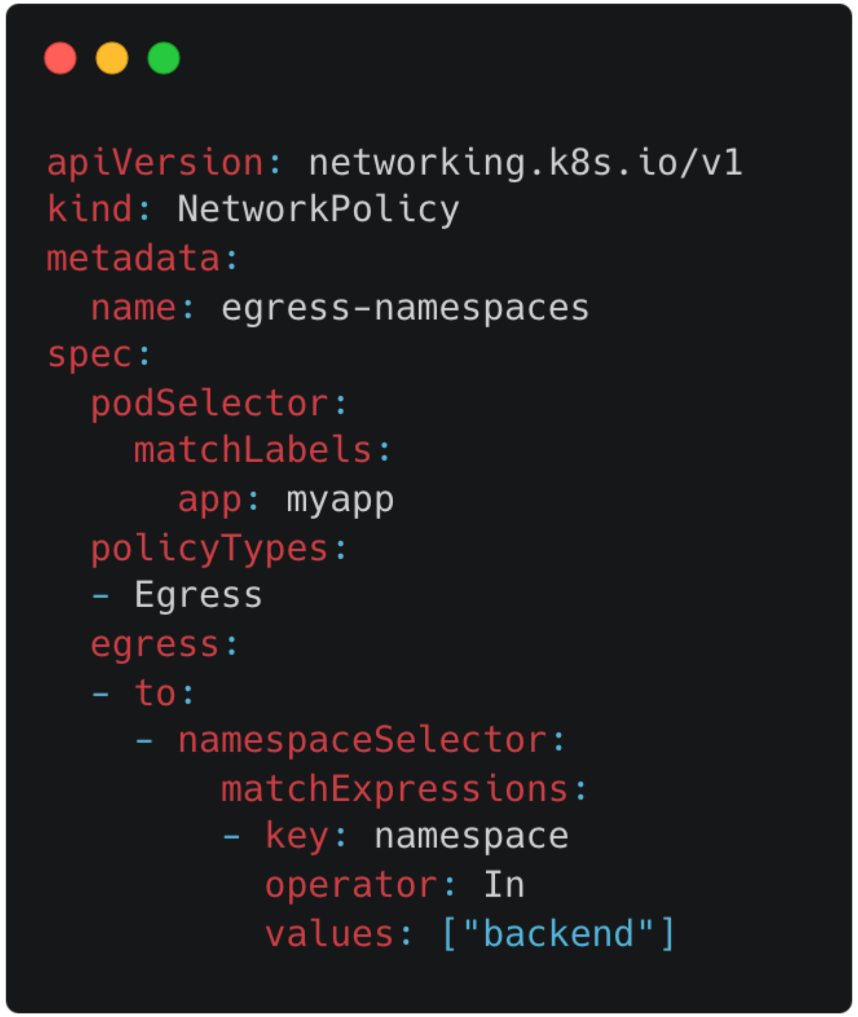
Policy: Allow all egress traffic



```yaml
---
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-all-egress
spec:
  podSelector: {}
  egress:
  - {}
  policyTypes:
  - Egress
```

kubernetes

Policy: Allow egress to namespace backend



```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: egress-namespaces
spec:
  podSelector:
    matchLabels:
      app: myapp
  policyTypes:
  - Egress
  egress:
  - to:
    - namespaceSelector:
        matchExpressions:
        - key: namespace
          operator: In
          values: ["backend"]
```

kubernetes

Network Plugins

Network Plugins

- These are software components that implement the Kubernetes networking model, allowing pods to communicate across the cluster seamlessly
- They manage the creation, maintenance, and teardown of network interfaces, implementing the Container Network Interface (CNI) specification.
- Network plugins must be compatible with the CNI specification to work within Kubernetes, enabling users to swap out implementations easily without modifying their applications

kubernetes

Network Plugins

- The choice of network plugin can significantly impact the performance and scalability of a Kubernetes cluster, affecting pod startup times, network latency, and throughput
- Some network plugins offer advanced features like network policy enforcement, multi-tenancy support, and integration with existing networking infrastructure, allowing for greater customization to meet specific security or performance requirements

Most Used Network Plugins

- **Calico**: Offers rich network policy enforcement and network fabric options, supporting both overlay and non-overlay networks, making it suitable for a wide range of environments.
- **Flannel**: Known for its simplicity and ease of use, Flannel is a very popular choice for basic networking needs, creating a simple overlay network for Kubernetes.
- **Cilium**: Utilizes BPF (Berkeley Packet Filter) to provide highly efficient networking and security policies, particularly suited for high-performance and security-sensitive environments.

kubernetes

Best Practices

Best Practices

- Start with “default-deny-all” to block everything and then start whitelisting needed connection for each application
- Understand rule evaluation since every rule is added to previous ones (rules are OR’ed and not AND’ed)
- Test your policies carefully and using several patterns: Allowed/blocked connections, External connectivity, Intra-namespace connectivity
- Good GUI for NetworkPolicies definitions: https://editor.networkpolicy.io/
- Good source to look into “recipes”: https://github.com/ahmetb/kubernetes-network-policy-recipes

kubernetes

Demo | Network Policies

Questions?

kubernetes