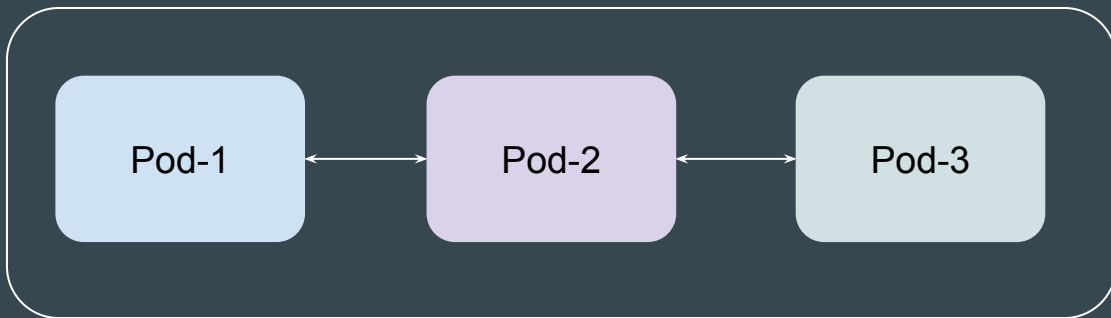


Overview of Network Policies

Understanding the Basics

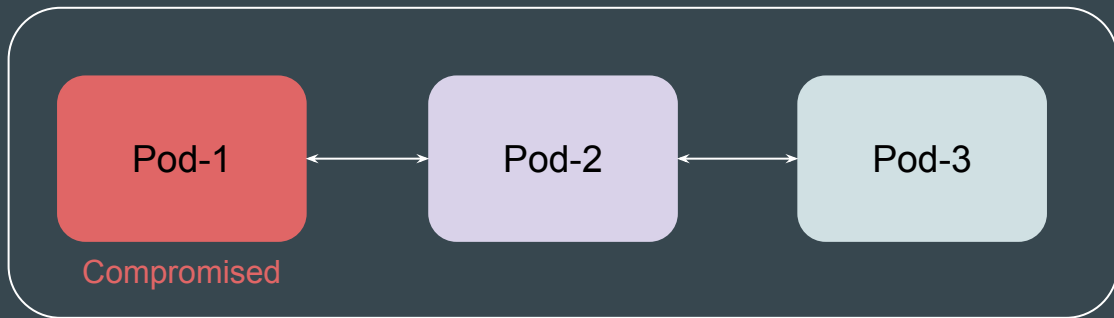
By default, Kubernetes **allows all traffic between pods within a cluster**. Network Policies help you lock down this open communication.



Kubernetes Cluster

Understanding the Challenge

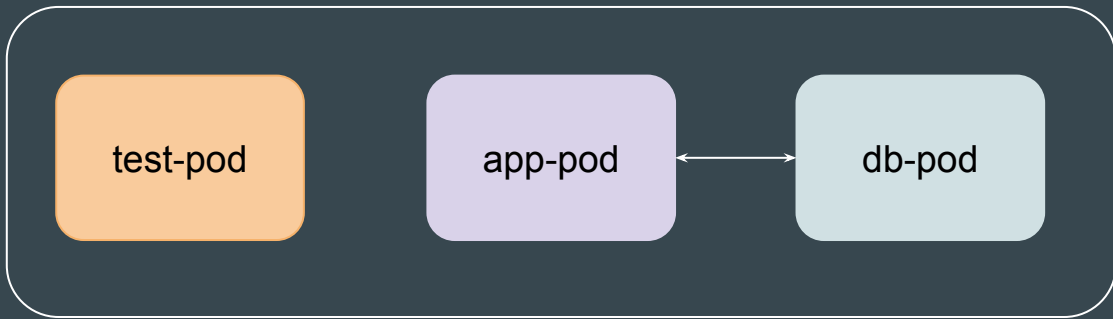
If a application inside any Pod gets compromised, attacker can essentially communicate with all other Pods easily over the network.



Kubernetes Cluster

Ideal Scenario

You only want Pods that have genuine requirement to connect to other pods to be able to communicate.



Kubernetes Cluster

Introducing Network Policies

Network Policies are a **mechanism for controlling network traffic** flow in Kubernetes clusters.

Source	Destination	Effect
app-pod	db-pod	Allow
test-pod	ALL	Deny

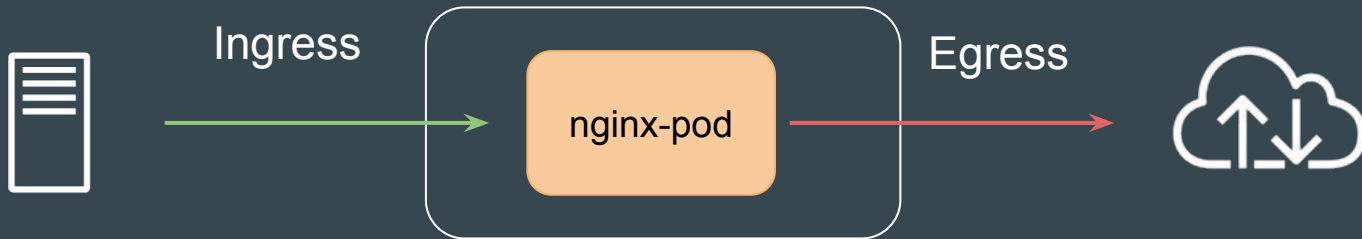
Network Policies



Types of Rules

There are two types of rules supported as part of Network policies:

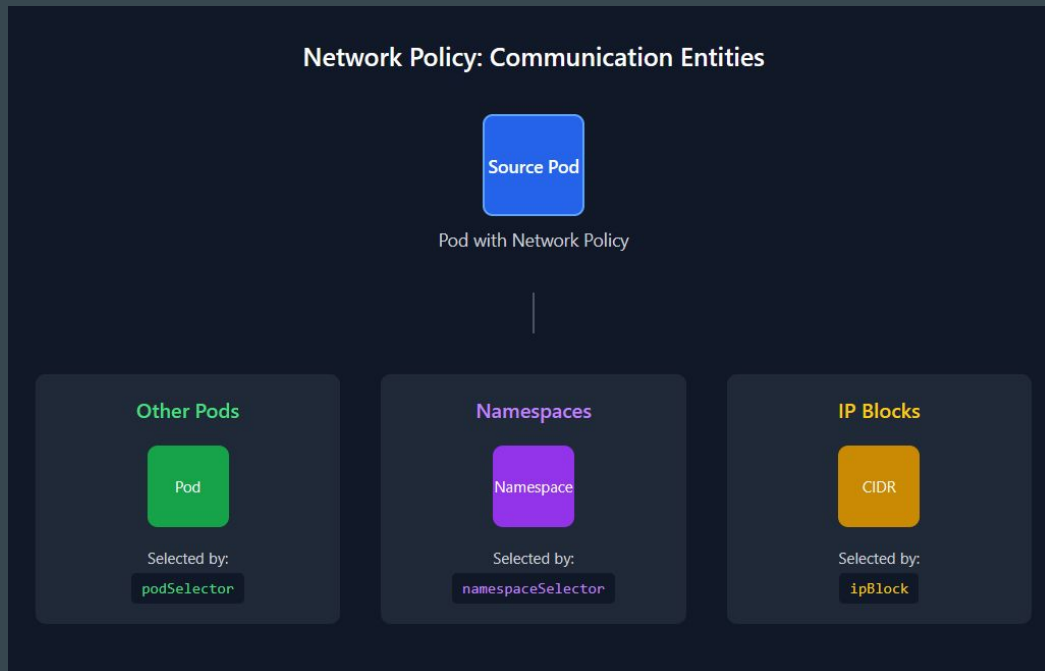
1. Ingress Rules (Inbound Rule)
2. Egress Rules (Outbound Rule)



Supported Filtering Entities

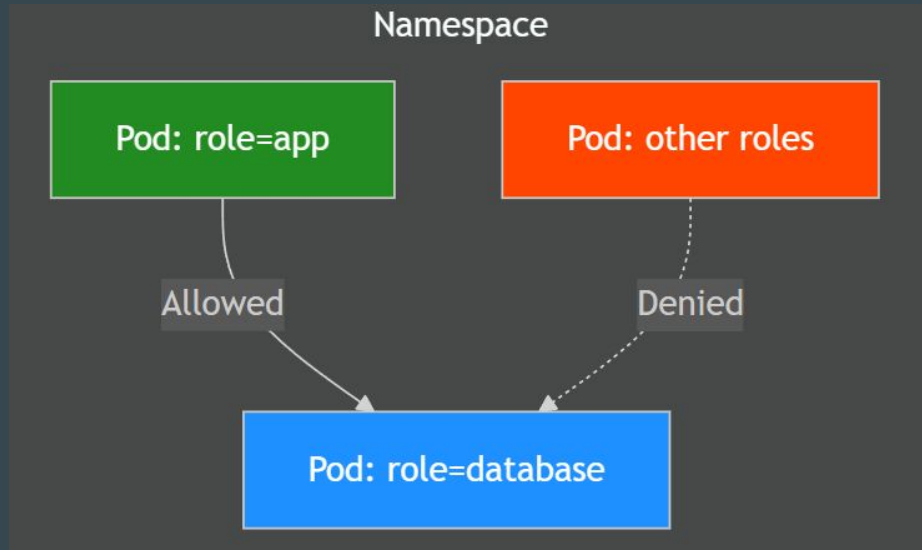
The entities that a Pod can communicate with are identified through a combination of the following three identifiers:

1. Other pods
2. Namespaces
3. IP Blocks



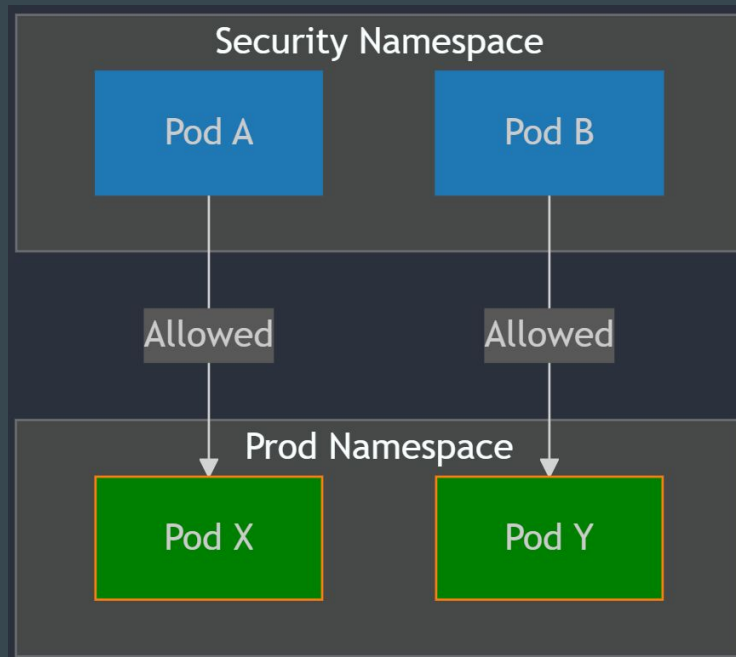
Example 1 - Pod Selector

Allow pods with label of role=app to connect to pods with labels of role=database



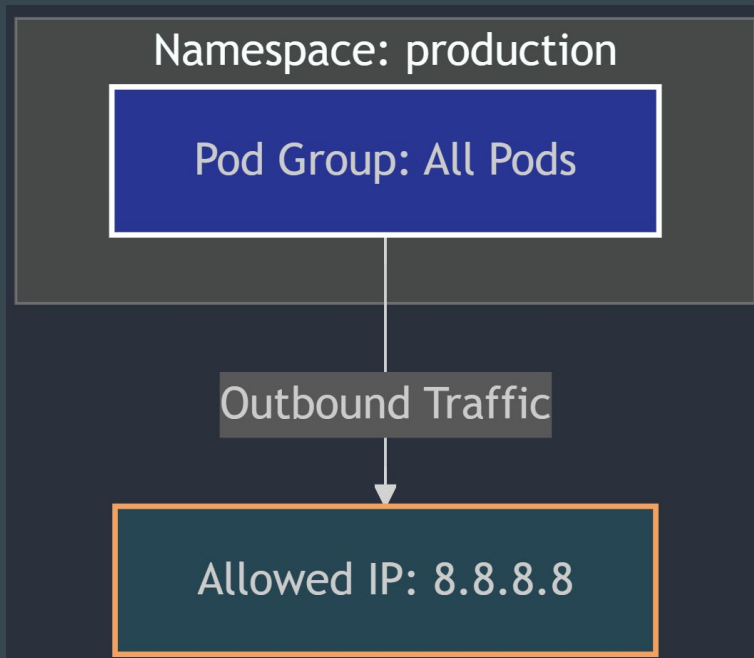
Example 2 - Namespace Selector

Allow Pods from Security namespace to connect to Pods in Prod namespace.



Example 3 - ipBlock

Allow Pods from production namespace to connect to only 8.8.8.8 IP address outbound.



Support for Network Policy

Not all Kubernetes network plugins (CNIs) support NetworkPolicy.

The ability to enforce NetworkPolicies is a feature that must be implemented by the CNI plugin

Some Network Plugins like Calico, Cilium, etc supports Network policy.

Some Network plugins like kubenet, Flannel does NOT support network Policy

Network Policy and Managed K8s Cluster

Most managed Kubernetes services (like AKS, EKS, GKE) come with a CNI that supports NetworkPolicy by default.

However, it's always a good idea to check the documentation for your specific service to confirm.

Structure of Network Policy

Sample Network Policy

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: demo-network-policy
spec:
  podSelector:
    matchLabels:
      env: production
  policyTypes:
    - Ingress
    - Egress
  ingress:
    - from:
        - podSelector:
            matchLabels:
              env: security
  egress:
    - to:
        - ipBlock:
            cidr: 8.8.8.8/32
```

Basic Mandatory Fields

As with all other Kubernetes config, a NetworkPolicy needs the following fields:

- apiVersion
- kind
- metadata

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: demo-network-policy
```

Contents of Spec

NetworkPolicy spec has all the information needed to define a particular network policy in the given namespace.

- podSelector
- policyTypes
- Ingress
- egress

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: demo-network-policy
spec:
  podSelector:
    matchLabels:
      env: production
  policyTypes:
    - Ingress
    - Egress
  ingress:
    - from:
        - podSelector:
            matchLabels:
              env: security
  egress:
    - to:
        - ipBlock:
            cidr: 8.8.8.8/32
```


1 - Pod Selector

Each NetworkPolicy includes a podSelector which selects the grouping of pods to which the policy applies.

The example network policy applies to all pods that has label of env=production

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: demo-network-policy
spec:
  podSelector:
    matchLabels:
      env: production
```

Point to Note

An empty podSelector selects all pods in the namespace.

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: demo-network-policy
spec:
  podSelector: {}
```

2 - Policy Types

Each NetworkPolicy includes a policyTypes list which may include either Ingress, Egress, or both.


The policyTypes field indicates whether or not the given policy applies to inbound traffic to selected pod, outbound traffic from selected pods, or both

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: demo-network-policy
spec:
  podSelector:
    matchLabels:
      env: production
  policyTypes:
    - Ingress
    - Egress
```

3 - Ingress

Inside ingress, you can use various combinations of podSelector, nameSpace selector etc to define the rules.

Inbound traffic for Pods with label of env=production will be allowed from pods with label env=security.



```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: demo-network-policy
spec:
  podSelector:
    matchLabels:
      env: production
  policyTypes:
    - Ingress
    - Egress
  ingress:
    - from:
        - podSelector:
            matchLabels:
              env: security
```

4 - Egress

Inside Egress rule, you can use various combinations of podSelector, namespaceSelector, ipBlock etc to define the rules.

Allow Outbound Traffic only to IP
address of 8.8.8.8 for Pods having
label of env=production



```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: demo-network-policy
spec:
  podSelector:
    matchLabels:
      env: production
  policyTypes:
    - Ingress
    - Egress
  ingress:
    - from:
        - podSelector:
            matchLabels:
              env: security
  egress:
    - to:
        - ipBlock:
            cidr: 8.8.8.8/32
```

Role of from and to

from	<p>Used in an Ingress rule.</p> <p>Specifies the sources of incoming traffic allowed to the selected pods.</p> <p>Sources can be other pods, namespaces, or IP blocks.</p>
to	<p>Used in an Egress rule.</p> <p>Specifies the destinations of outgoing traffic allowed from the selected pods.</p> <p>Destinations can be other pods, namespaces, or IP blocks</p>

Practical - Network Policies

Example 1 - Block All Ingress and Egress

Since no specific rules are defined for ingress or egress, Kubernetes denies all traffic by default

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny-all
  namespace: production
spec:
  podSelector: {}
  policyTypes:
    - Ingress
    - Egress
```



Points to Note - podSelector

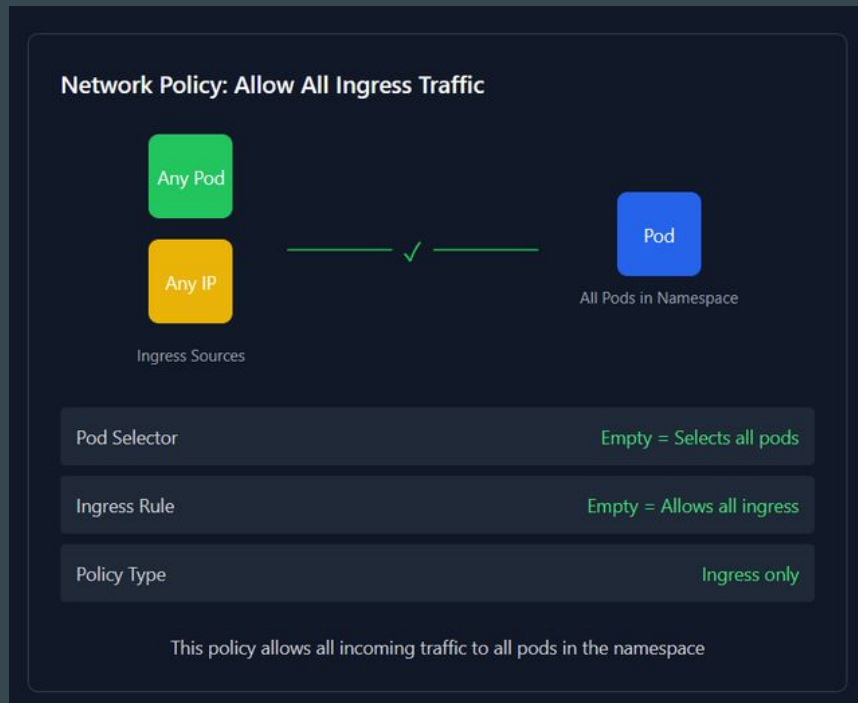
```
podSelector { }
```

This means the policy applies to all pods in the namespace because the selector is empty (matches all pods).

Example 2 - Allow Ingress Traffic

This policy allows all incoming traffic (ingress) to the selected pods.

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



Points to Note

ingress:

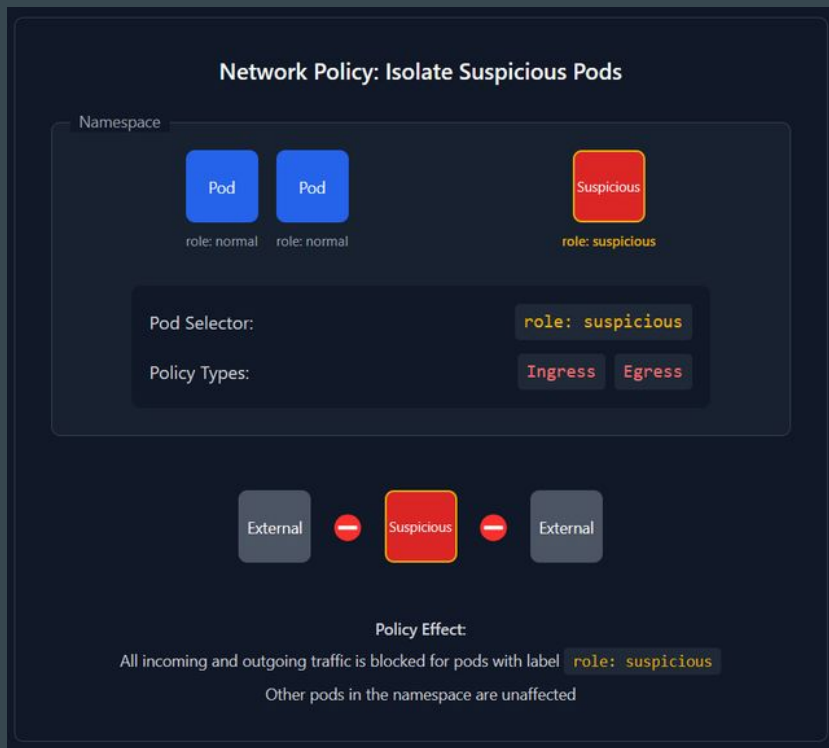
- {}

The empty {} means that there are no restrictions on the source of the traffic (any source is allowed).

Example 3 - Isolate Suspicious Pod

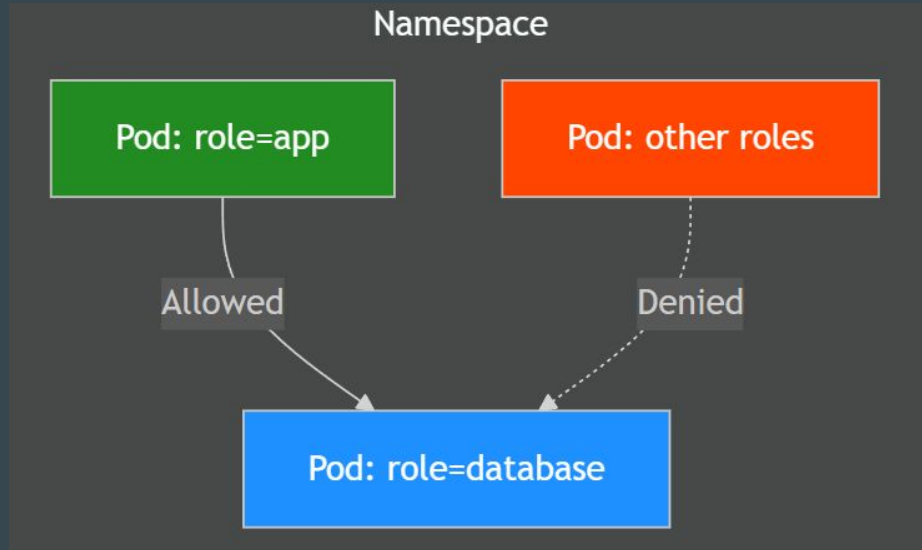
This policy blocks all ingress and egress access to pod with role=suspicious.

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: suspicious-pod
spec:
  podSelector:
    matchLabels:
      role: suspicious
  policyTypes:
    - Ingress
    - Egress
```



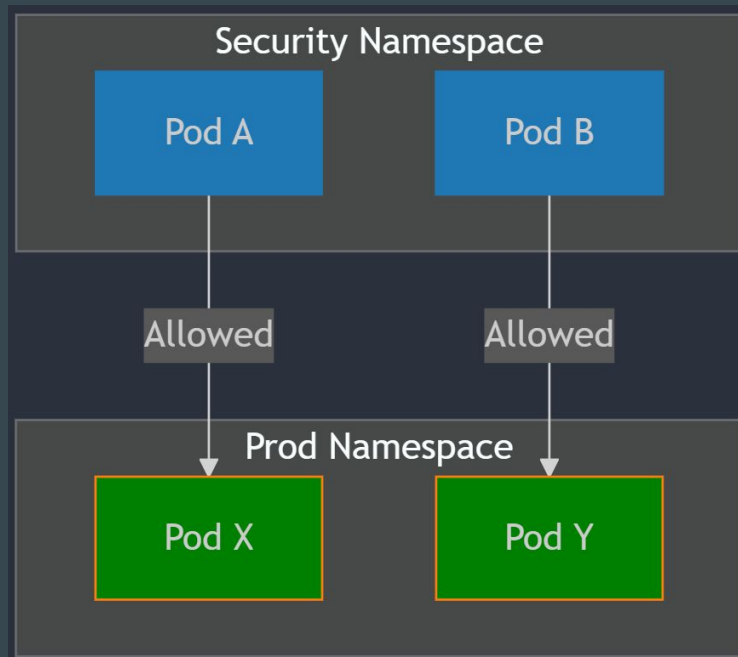
Example 4 - PodSelector

Allow pods with label of role=app to connect to pods with labels of role=database



Example 5 - Namespace Selector

Allow Pods from Security namespace to connect to Pods in Prod namespace.

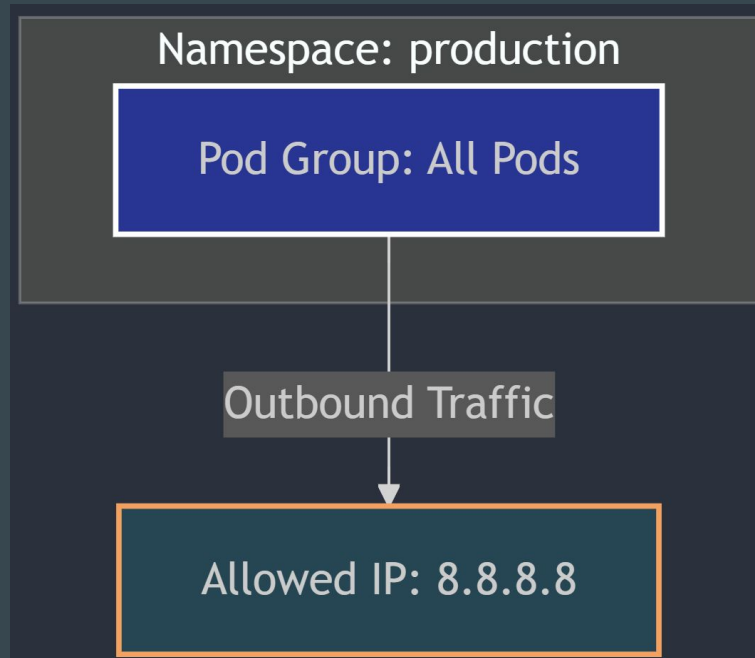


Example 5 - Reference Code

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: namespace-esselector
  namespace: production
spec:
  podSelector: {}
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
                kubernetes.io/metadata.name: security
  policyTypes:
    - Ingress
```

Example 6 - ipBlock

Allow Pods from production namespace to connect to only 8.8.8.8 IP address outbound.



Example 6 - Reference Code

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: outbound-8888
spec:
  podSelector: {}
  egress:
    - to:
        - ipBlock:
            cidr: 8.8.8.8/32
  policyTypes:
    - Egress
```

Network Policies - Except, Port and Protocol

Except

The **except** field in a Kubernetes NetworkPolicy allows you to define exceptions to a broader rule.

Following policy allows ingress for cidr range of 172.17.0.0/16 except the range of 172.17.1.0/24

```
ingress:
- from:
  - ipBlock:
      cidr: 172.17.0.0/16
      except:
        - 172.17.1.0/24
```

Reference Screenshot - Entire Policy

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: except-ingress
spec:
  podSelector:
    matchLabels:
      role: database
  ingress:
    - from:
        - ipBlock:
            cidr: 172.17.0.0/16
            except:
              - 172.17.1.0/24
  policyTypes:
    - Ingress
```

Ports and Protocol

When writing a NetworkPolicy, you can target a port or range of ports.

```
egress:
  - to:
    - ipBlock:
        cidr: 10.0.0.0/24
  ports:
    - protocol: TCP
      port: 32000
      endPort: 32768
```

Reference Screenshot - Entire Policy

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: multi-port-egress
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
    - Egress
  egress:
    - to:
        - ipBlock:
            cidr: 10.0.0.0/24
      ports:
        - protocol: TCP
          port: 32000
          endPort: 32768
```