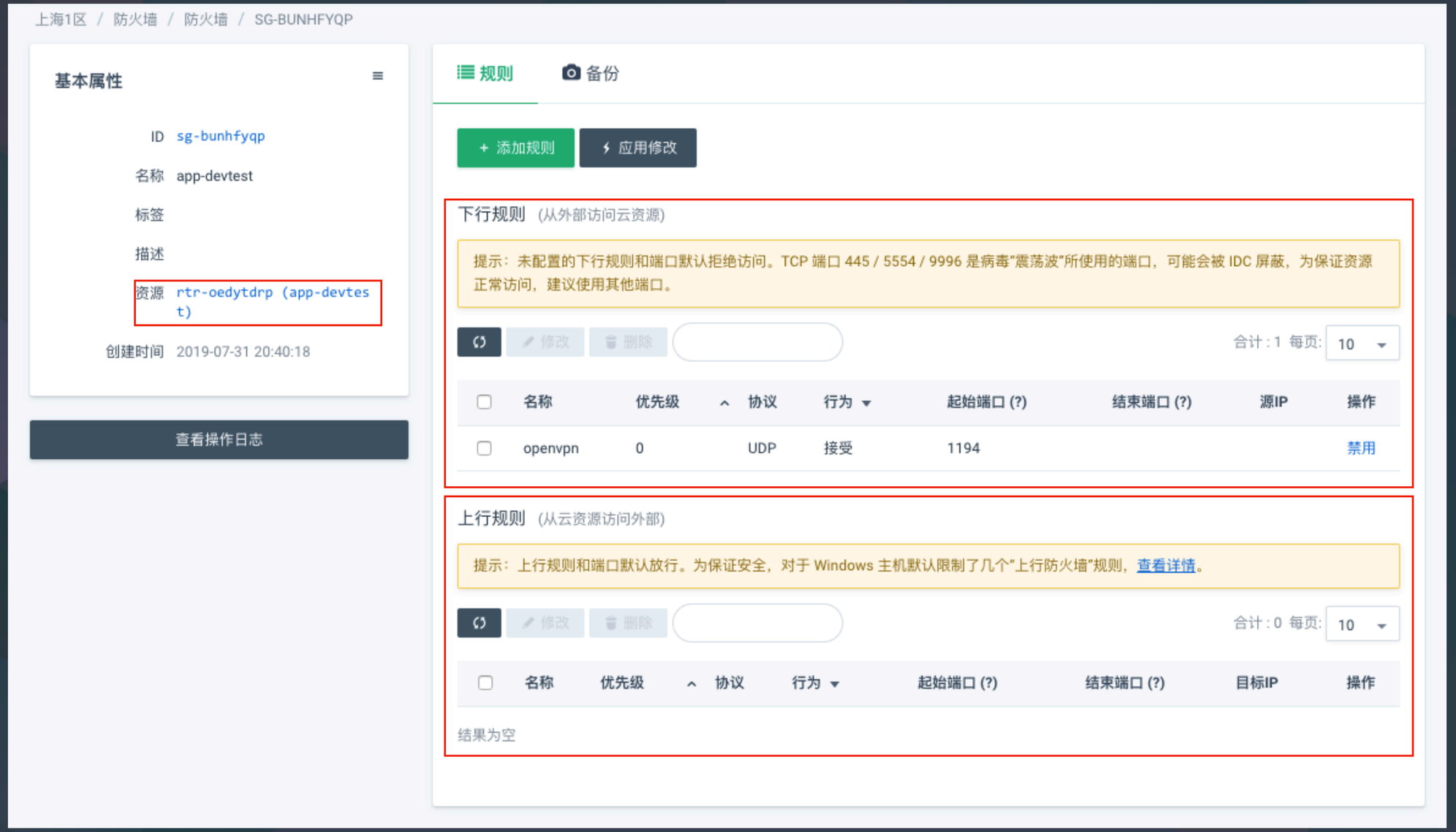
NetworkPolicy 最佳实践

Agenda

- 什么是 NetworkPolicy
- 如何编写 NetworkPolicy
- NetworkPolicy 如何运行
- 最佳实践

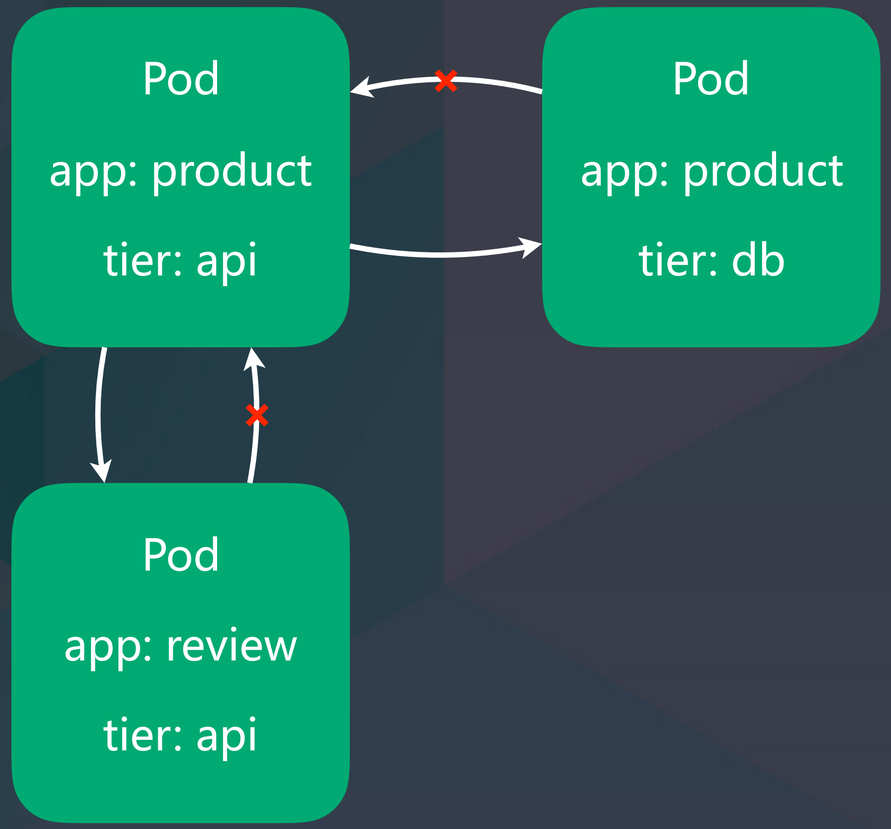
QINGCLOUD

什么是 NetworkPolicy



NetworkPolicy 就是云原生语境下的防火墙

QINGCLOUD



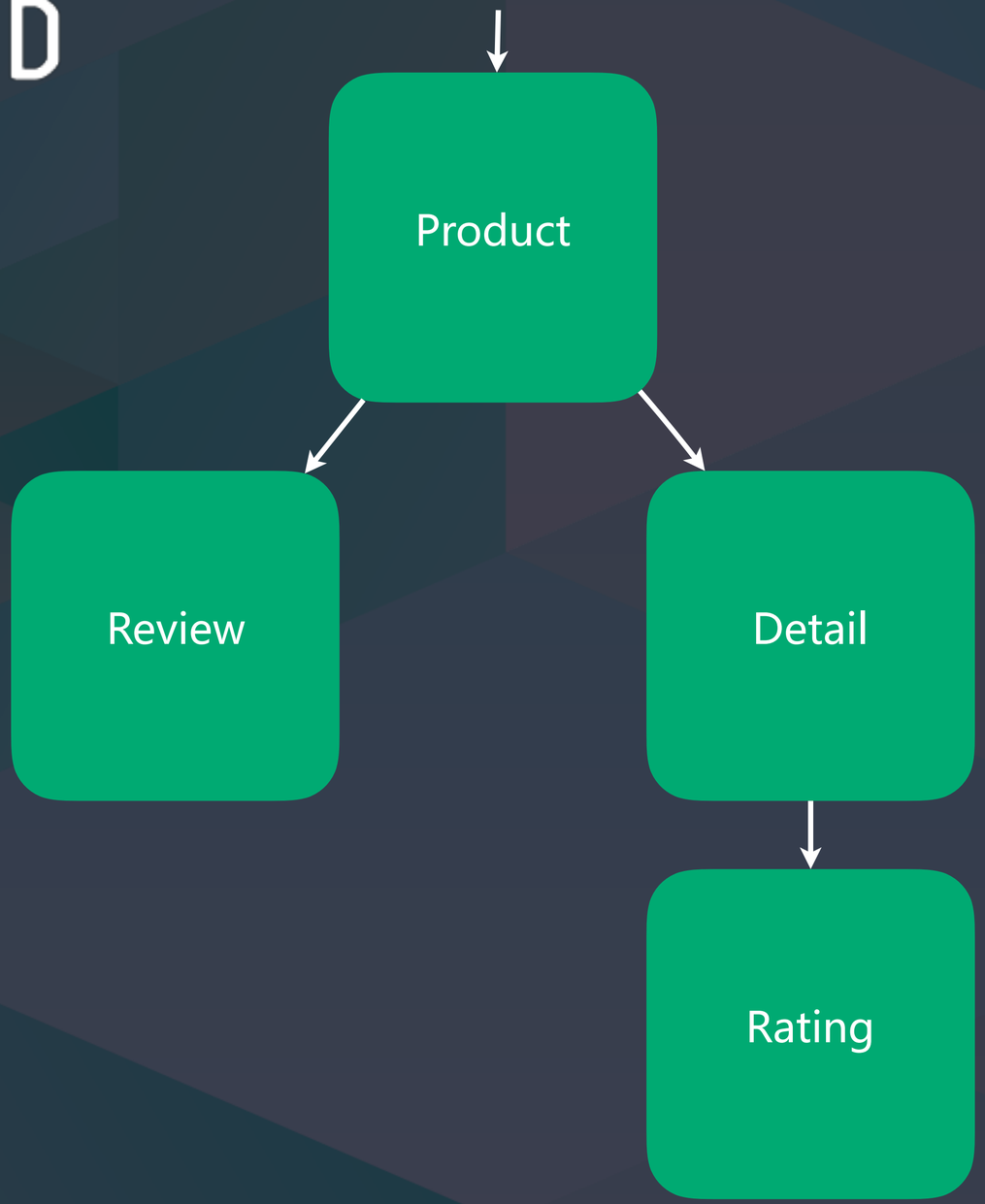
NetworkPolicy 控制 Pod 的入口、出口流量

QINGCLOUD

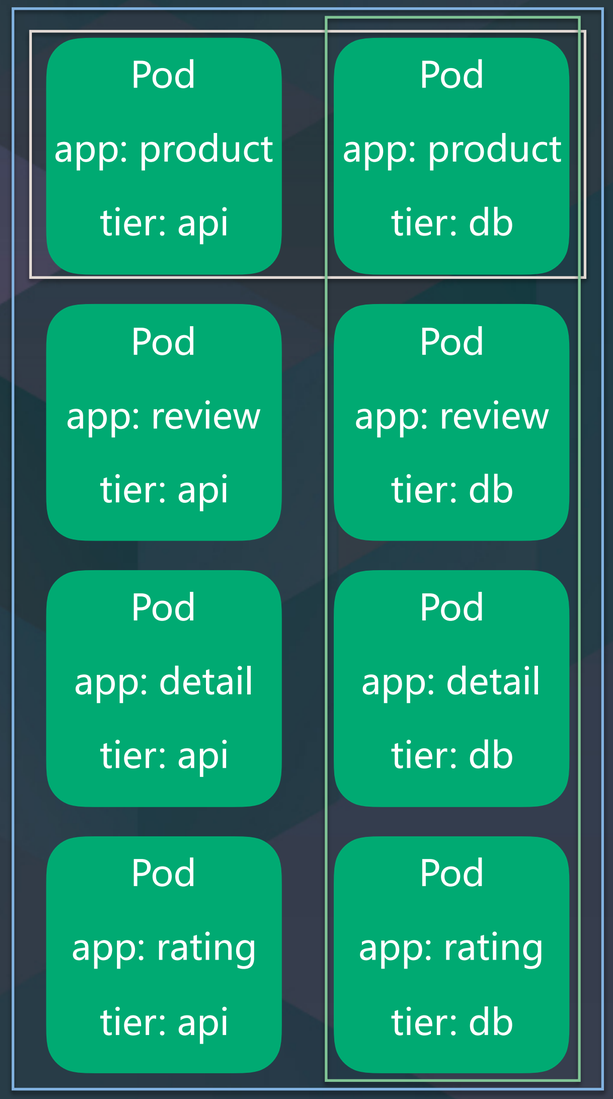
QINGCLOUD

如何编写 NetworkPolicy

QINGCLOUD



使用 Label Selector 机制选择 Pod 资源




```
PodSelector:
  MatchLabels: app=product
```

```
PodSelector:
  MatchLabels: tier=db
```

```
PodSelector:
  MatchLabels: {}
```

QINGCLOUD

NetworkPolicy 的构成要素

- 匹配哪些 Pod
- 入口流量的规则，表示谁允许访问这些 Pod
- 出口流量的规则，表示这些 Pod 允许访问谁

Sample

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

QINGCLOUD

一些基础规则

- 如果 Pod 没有被 NetworkPolicy 匹配到，那么它的流量是被允许的
- 如果 Pod 被 NetworkPolicy 匹配到，但是没有出口/入口规则被匹配到，那么它的出口/入口流量是被禁止的
- 你只能指定规则来允许流量通行，而不能直接禁止流量通行

例子1：禁止指定 Pod 入口流量


```yaml
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: review-deny-all
  namespace: myproject
spec:
  podSelector:
    matchLabels:
      app: review
  ingress: []
```

✨知识点✨
空数组表示不匹配任何规则

QINGCLOUD

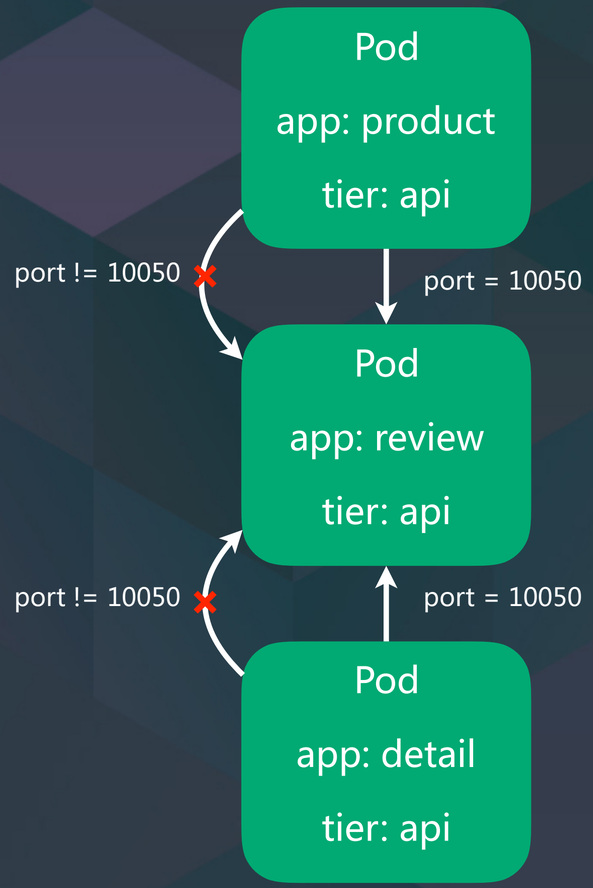
例子2：允许某些流量


```yaml
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: review-deny-all
  namespace: myproject
spec:
  podSelector:
    matchLabels:
      app: review
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: product
          tier: api
```

✨知识点✨
matchLabels内部是AND关系

QINGCLOUD

例子3：开放被选择 Pod 的指定端口




```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: ...
  namespace: ...
spec:
  podSelector:
    matchLabels:
      app: review
  ingress:
  - from:
    - ports:
      - port: 10050
```

✨知识点✨
使用 ContainerPort，而不是使用Service中定义的Port

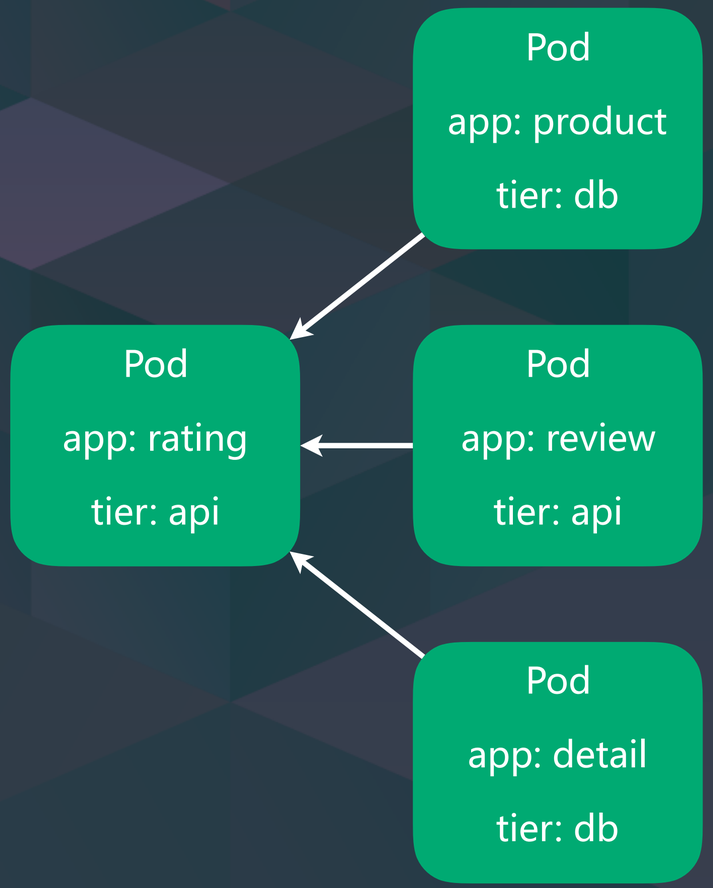
QINGCLOUD

一些基础规则

- 如果 Pod 没有被 NetworkPolicy 匹配到，那么它的流量是被允许的
- 如果 Pod 被 NetworkPolicy 匹配到，但是没有出口/入口规则被匹配到，那么它的出口/入口流量是被禁止的
- 你只能指定规则来允许流量通行，而不能直接禁止流量通行
- NetworkPolicy 中的 Rule 之间的匹配逻辑是 OR

QINGCLOUD

例子4：多个规则和选择器组合

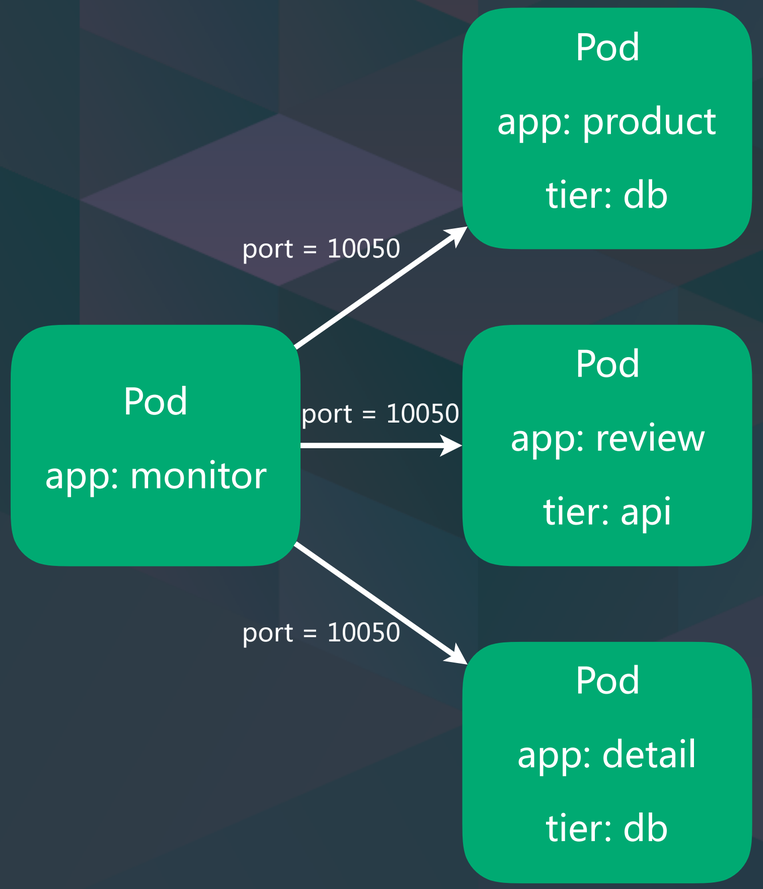



```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: ...
  namespace: ...
spec:
  podSelector: ...
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: product
          tier: db
    - podSelector:
        matchLabels:
          app: review
          tier: api
  - from:
    - podSelector:
        matchLabels:
          app: detail
          tier: db
```

✨知识点✨
蓝色的 selector 和绿色的 rule 之间的匹配逻辑都是OR

QINGCLOUD

例子5：为监控服务开放所有 Pod 的 10050 端口




```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: ...
  namespace: ...
spec:
  podSelector: {}
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: monitor
    ports:
    - port: 10050
```

✨知识点✨
这里只会匹配相同的 namespace

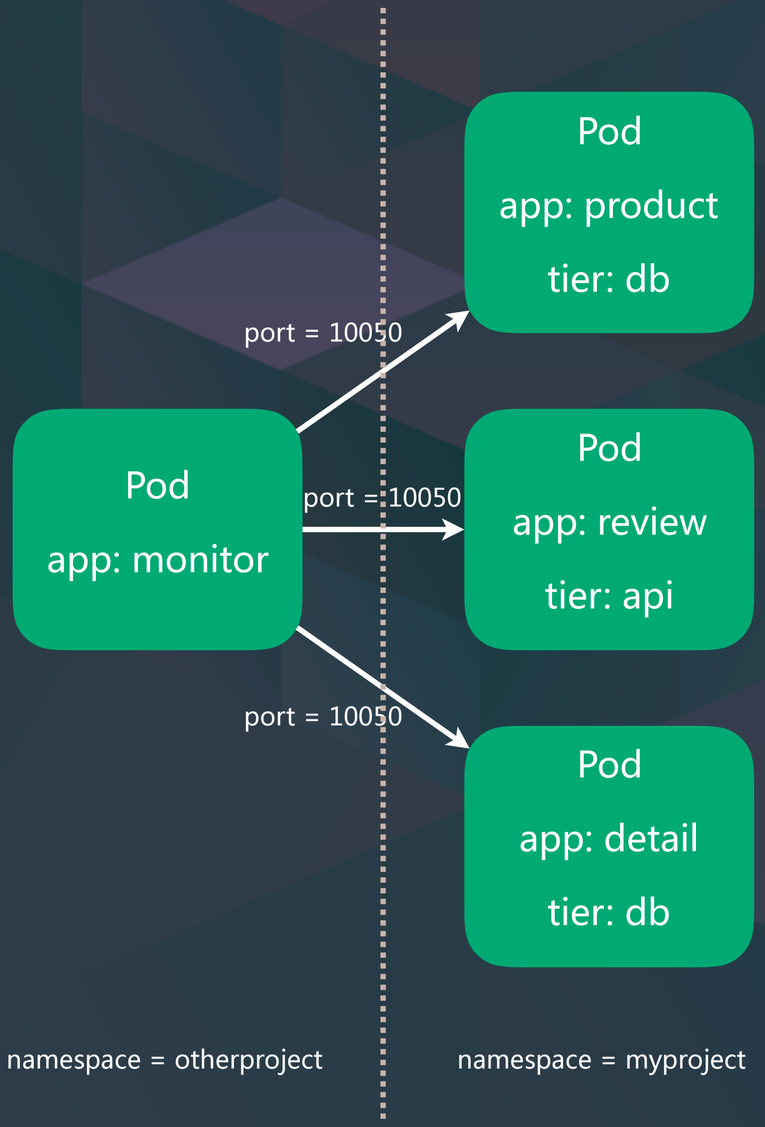
QINGCLOUD

一些基础规则

- 如果 Pod 没有被 NetworkPolicy 匹配到，那么它的流量是被允许的
- 如果 Pod 被 NetworkPolicy 匹配到，但是没有出口/入口规则被匹配到，那么它的出口/入口流量是被禁止的
- 你只能指定规则来允许流量通行，而不能直接禁止流量通行
- NetworkPolicy 中的 Rule 之间的匹配逻辑是 OR
- NetworkPolicy 默认的作用域是 Pod 所在的 Namespace

QINGCLOUD

例子5：允许来自其他 Namespace 的流量



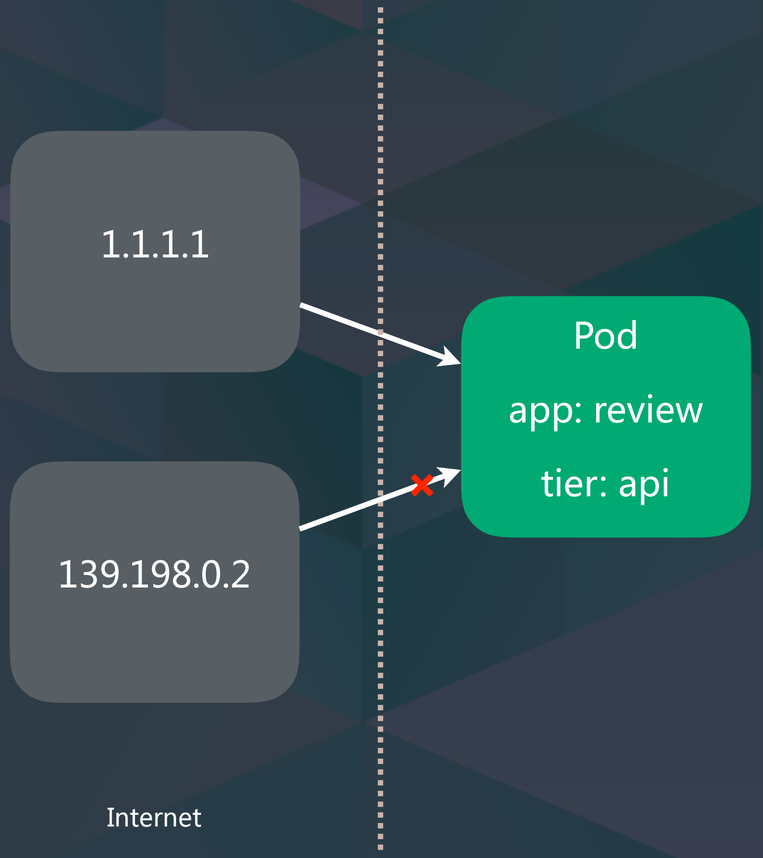

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: ...
  namespace: myproject
spec:
  podSelector: ...
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          environment: production
---
apiVersion: v1
kind: Namespace
metadata:
  name: otherproject
  labels:
    environment: production
```

QINGCLOUD

namespaceSelector 和 podSelector 都是面向集群内部的，如果我要限制外部访问应该怎么办？

QINGCLOUD

例子6：限制来自某些外部 IP 地址




```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: ...
  namespace: ...
spec:
  podSelector: ...
  ingress:
  - from:
    - ipBlock:
        cidr: 0.0.0.0/0
        except:
        - 139.198.0.0/16
```

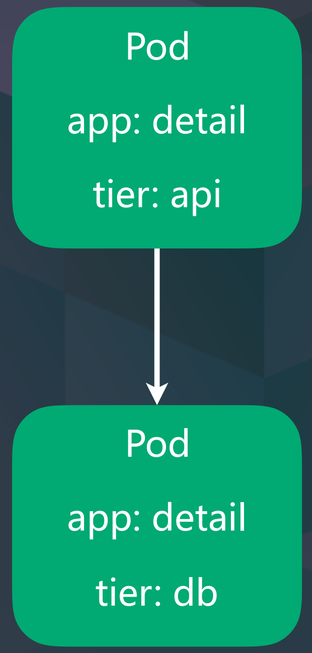
QINGCLOUD

NetworkPolicy 的构成要素

- 匹配哪些 Pod
- 入口流量的规则，表示谁允许访问这些 Pod
- 出口流量的规则，表示这些 Pod 允许访问谁

QINGCLOUD

例子7：允许某些出口流量

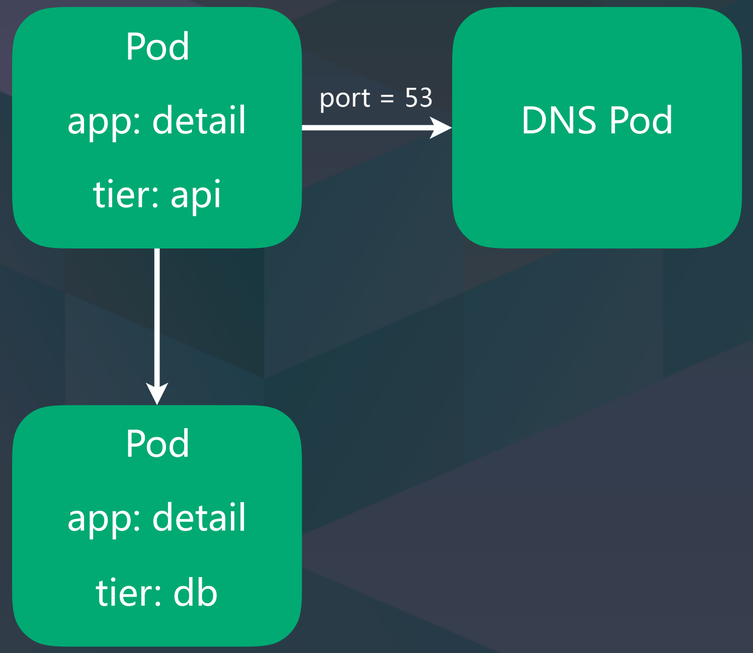



```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: ...
  namespace: ...
spec:
  podSelector:
    matchLabels:
      tier: api
  policyTypes:
  - Egress
  egress:
  - to:
    - podSelector:
        matchLabels:
          tier: db
```

```
# curl http://database
curl: (6) Could not resolve host: database
```

QINGCLOUD

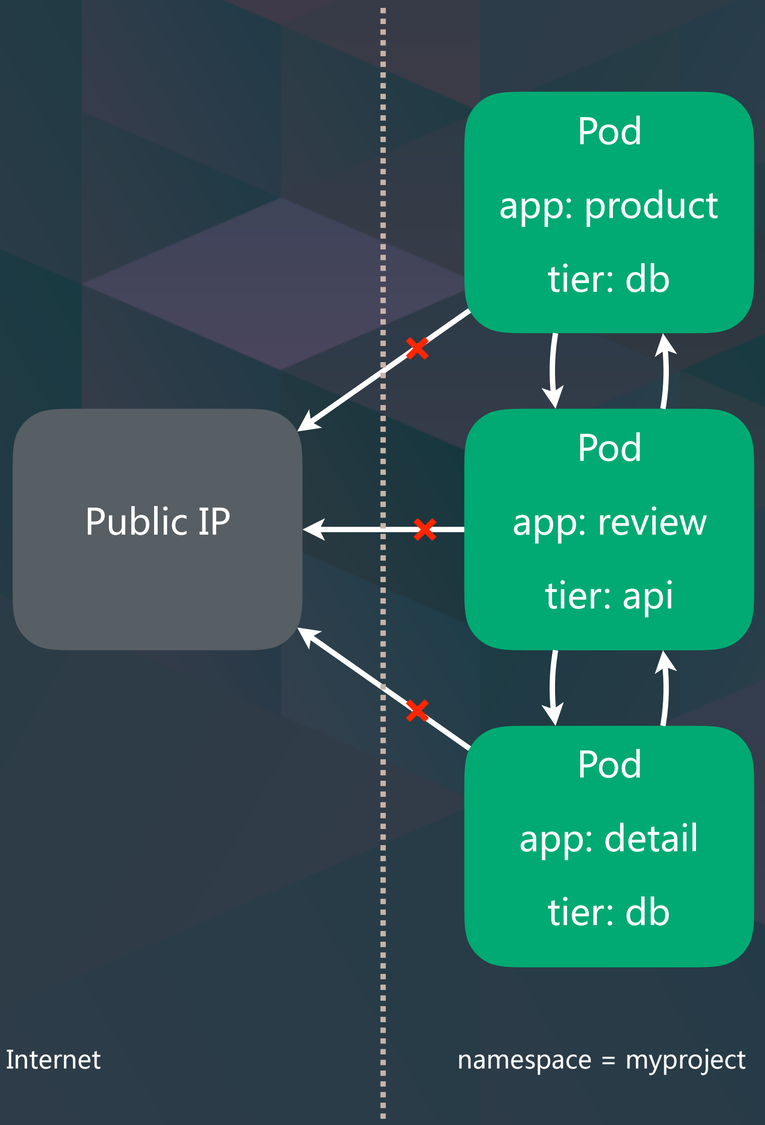
例子7：允许某些出口流量




```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: ...
  namespace: ...
spec:
  podSelector:
    matchLabels:
      tier: api
  policyTypes:
  - Egress
  egress:
  - to:
    - podSelector:
        matchLabels:
          tier: db
  - to:
    - namespaceSelector: {}
    ports:
    - protocol: UDP
      port: 53
    - protocol: TCP
      port: 53
```

QINGCLOUD

例子8：允许访问集群内部，禁止访问集群外部




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

✨建议✨
用贴切的名字定义
NetworkPolicy 的内容

QINGCLOUD

QINGCLOUD

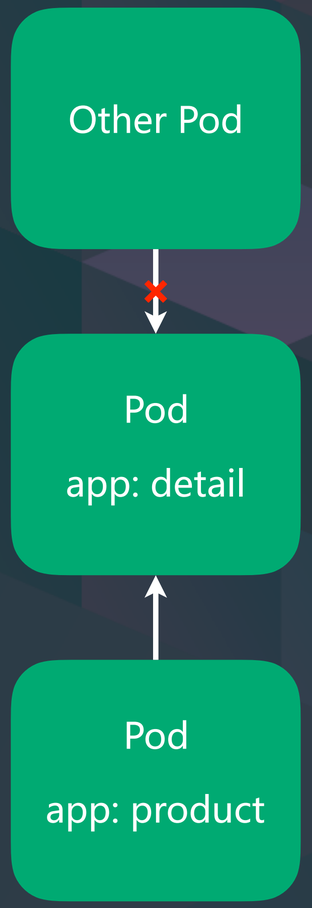
NetworkPolicy 如何运行

常见的支持 NetworkPolicy 的 CNI 插件

- Calico
- Cilium
- Kube-router
- Romana
- Weave Net

QINGCLOUD

以 Calico 为例，分析基于 iptables 的实现




```
root@i-d5n5dqkp:~# kubectl get node -o wide
NAME         STATUS   ROLES       AGE   VERSION   INTERNAL-IP    EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION      CONTAINER-RUNTIME
i-m0yo3jaj   Ready    master      14d   v1.13.5   192.168.2.11   <none>        Ubuntu 18.04.2 LTS   4.15.0-55-generic   docker://18.6.2
i-qpnavoby   Ready    node_perf   14d   v1.13.5   192.168.2.14   <none>        Ubuntu 18.04.2 LTS   4.15.0-55-generic   docker://18.6.2
i-yudgee7v   Ready    node_perf   14d   v1.13.5   192.168.2.13   <none>        Ubuntu 18.04.2 LTS   4.15.0-55-generic   docker://18.6.2
```

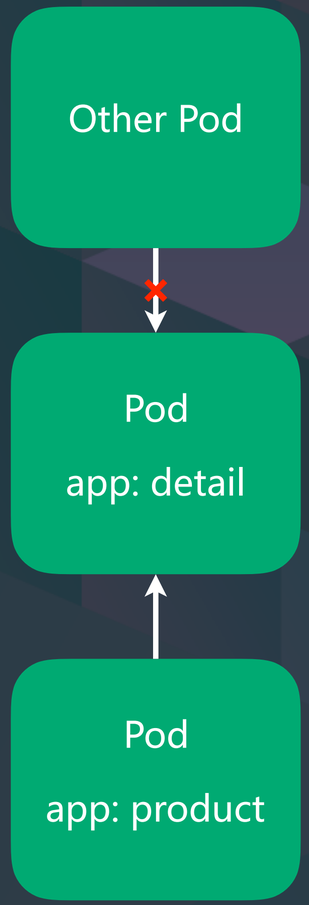
```
root@i-d5n5dqkp:~# kubectl get pod -o wide -n kube-system | grep calico
calico-node-67j6d                       2/2     Running   0          14d   192.168.2.11   i-m0yo3jaj   <none>           <none>
calico-node-dv57w                       2/2     Running   0          14d   192.168.2.13   i-yudgee7v   <none>           <none>
calico-node-kjzxj                       2/2     Running   0          14d   192.168.2.14   i-qpnavoby   <none>           <none>
```

```
root@i-d5n5dqkp:~# kubectl run detail --image=nginx --labels app=detail --expose --port 80
kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be removed in a future version. Use kubectl run --generator=run-pod/v1 or kubectl create instead.
service/detail created
deployment.apps/detail created
```

```
root@i-d5n5dqkp:~# kubectl run test-$RANDOM --generator=run-pod/v1 --rm -i -t --image=alpine -- sh
If you don't see a command prompt, try pressing enter.
/ # wget -qO- --timeout=2 http://detail
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
```

QINGCLOUD

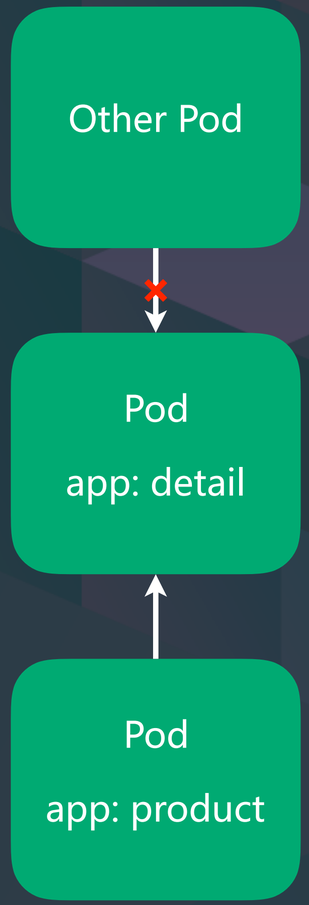
以 Calico 为例，分析基于 iptables 的实现




```
root@i-d5n5dqkp:~# cat detail-allow-product.yaml
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: detail-allow-product
spec:
  podSelector:
    matchLabels:
      app: detail
  ingress:
  - from:
      - podSelector:
          matchLabels:
            app: product
root@i-d5n5dqkp:~#
root@i-d5n5dqkp:~# kubectl apply -f detail-allow-product.yaml
networkpolicy.networking.k8s.io/detail-allow-product created
root@i-d5n5dqkp:~#
root@i-d5n5dqkp:~# kubectl run test-$RANDOM --generator=run-pod/v1 --rm -i -t --image=alpine -- sh
If you don't see a command prompt, try pressing enter.
/ # wget -qO- --timeout=2 http://detail
wget: download timed out
/ # exit
Session ended, resume using 'kubectl attach test-16506 -c test-16506 -i -t' command when the pod is running
pod "test-16506" deleted
root@i-d5n5dqkp:~#
root@i-d5n5dqkp:~# kubectl run test-$RANDOM --generator=run-pod/v1 --rm -i -t --image=alpine --labels app=product -- sh
If you don't see a command prompt, try pressing enter.
/ # wget -qO- --timeout=2 http://detail
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
```

QINGCLOUD

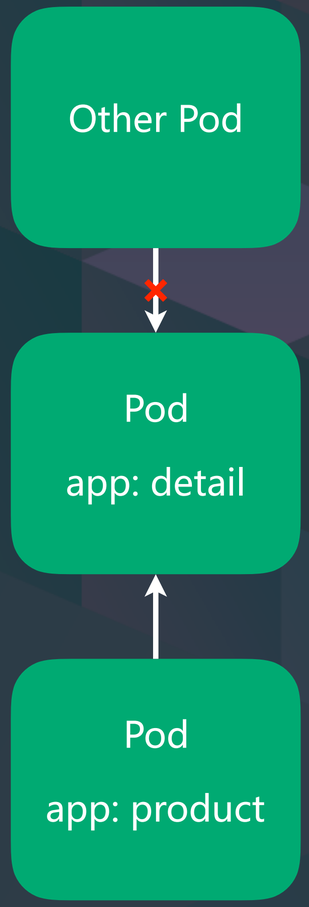
以 Calico 为例，分析基于 iptables 的实现




```
root@i-tf0mcida:~# kubectl get pod -o wide
NAME                      READY   STATUS    RESTARTS   AGE   IP           NODE         NOMINATED NODE   READINESS GATES
detail-b54f75fd6-pqmg5    1/1     Running   0          28m   10.10.2.74   i-yudgee7v   <none>           <none>
test-28305                1/1     Running   0          12m   10.10.1.72   i-qpnavoby   <none>           <none>
root@i-tf0mcida:~# ssh i-yudgee7v
root@i-yudgee7v:~# ip r | grep 10.10.2.74
10.10.2.74 dev cali97ca93548f9 scope link
root@i-yudgee7v:~# iptables-save -c -t filter | grep cali-tw-cali97ca93548f9
:cali-tw-cali97ca93548f9 - [0:0]
[16:989] -A cali-to-wl-dispatch-9 -o cali97ca93548f9 -m comment --comment "cali:WP6rAnhTeUrMpd9j" -g cali-tw-cali97ca93548f9
[5:329] -A cali-tw-cali97ca93548f9 -m comment --comment "cali:tlJOCSJnM0eBfmrB" -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
[0:0] -A cali-tw-cali97ca93548f9 -m comment --comment "cali:2hLKFThQ3mHCWM_9" -m conntrack --ctstate INVALID -j DROP
[11:660] -A cali-tw-cali97ca93548f9 -m comment --comment "cali:PXJ80Gx4msRAnTQU" -j MARK --set-xmark 0x0/0x10000
[3:180] -A cali-tw-cali97ca93548f9 -m comment --comment "cali:6-m-X3jF3UDl55xt" -m comment --comment "Start of policies" -j MARK --set-xmark 0x0/0x20000
[3:180] -A cali-tw-cali97ca93548f9 -m comment --comment "cali:OJaJfV5NOjqRwn6p" -m mark --mark 0x0/0x20000 -j cali-pi-_LogYC2-9R_dvLQPVucW
[1:60] -A cali-tw-cali97ca93548f9 -m comment --comment "cali:yuWhwflcZWyY9GiI" -m comment --comment "Return if policy accepted" -m mark --mark 0x10000/0x10000 -j RETURN
[2:120] -A cali-tw-cali97ca93548f9 -m comment --comment "cali:7-yMuIXl7QkJbUWO" -m comment --comment "Drop if no policies passed packet" -m mark --mark 0x0/0x20000 -j DROP
[0:0] -A cali-tw-cali97ca93548f9 -m comment --comment "cali:fQ2VJfLKuZoXYbov" -j cali-pri-kns.default
[0:0] -A cali-tw-cali97ca93548f9 -m comment --comment "cali:2b9dlDfWwDKb4HCW" -m comment --comment "Return if profile accepted" -m mark --mark 0x10000/0x10000 -j RETURN
```

QINGCLOUD

以 Calico 为例，分析基于 iptables 的实现




```
root@i-yudgee7v:~# iptables-save -c -t filter | grep cali-pi-_LogYC2-9R_dvLQPVucW
:cali-pi-_LogYC2-9R_dvLQPVucW - [0:0]
[1:60] -A cali-pi-_LogYC2-9R_dvLQPVucW -m comment --comment "cali:-MIR5_nZm2PMzZqh" -m set --match-set cali40s:YhdaWjHFD7z6MtbO2DHTeeX src -j MARK --set-xmark 0x10000/0x10000
[1:60] -A cali-pi-_LogYC2-9R_dvLQPVucW -m comment --comment "cali:DzWz7CvTWAofjsKu" -m mark --mark 0x10000/0x10000 -j RETURN
[3:180] -A cali-tw-cali97ca93548f9 -m comment --comment "cali:OJaJfV5NOjqRwn6p" -m mark --mark 0x0/0x20000 -j cali-pi-_LogYC2-9R_dvLQPVucW
```

```
root@i-yudgee7v:~# ipset list cali40s:YhdaWjHFD7z6MtbO2DHTeeX
Name: cali40s:YhdaWjHFD7z6MtbO2DHTeeX
Type: hash:net
Revision: 6
Header: family inet hashsize 1024 maxelem 1048576
Size in memory: 408
References: 3
Number of entries: 1
Members:
10.10.1.72
```

```
root@i-yudgee7v:~# logout
Connection to i-yudgee7v closed.
```

```
root@i-tf0mcida:~# kubectl get pod -o wide
NAME                     READY   STATUS    RESTARTS   AGE   IP           NODE         NOMINATED NODE   READINESS GATES
detail-b54f75fd6-pqmg5   1/1     Running   0          32m   10.10.2.74   i-yudgee7v   <none>           <none>
test-28305               1/1     Running   0          16m   10.10.1.72   i-qpnavoby   <none>           <none>
```

```
root@i-tf0mcida:~# kubectl get pod -o wide
NAME                     READY   STATUS    RESTARTS   AGE   IP           NODE         NOMINATED NODE   READINESS GATES
detail-b54f75fd6-pqmg5   1/1     Running   0          40m   10.10.2.74   i-yudgee7v   <none>           <none>
test-28305               1/1     Running   0          24m   10.10.1.72   i-qpnavoby   <none>           <none>
root@i-tf0mcida:~# kubectl describe pod test-28305 | grep Labels
Labels:             app=product
```

QINGCLOUD

QINGCLOUD

最佳实践

创建一条 default-deny-all 规则

- 禁止 Namespace 内所有的出口和入口流量
- 针对对需要的应用或者 Namespace 或者外网启用白名单

QINGCLOUD

网络策略

- pod选择器, 表示当前网络策略应用在哪些pod上, "选择器语法"
- 入口策略组, 表示允许匹配到入口策略的pod访问被上面的pod选择器选中的pod, 匹配逻辑为or
  - 入口策略, "网络策略规则语法"
  - ...
- 出口策略组, 表示允许匹配到出口策略的pod被上面pod选择器选中的pod所访问, 匹配逻辑为or
  - 出口策略, "网络策略规则语法"
  - ...

熟悉 NetworkPolicy 的书写语法

选择器语法

下列匹配方式选其一

- 匹配label, key-value形式, 匹配逻辑为and
  - key: value
  - ...
- 匹配表达式, 匹配逻辑为and
  - 表达式, key [In/NotIn/Exists/DoesNotExist] [values]
  - ...

网络策略规则语法

- 端口规则组, 匹配逻辑为or
  - 端口, 匹配逻辑为and
    - 端口号
    - 协议
  - ...
- 匹配规则组, 匹配逻辑为or
  - 匹配规则, 匹配逻辑为and
    - pod选择器, "选择器语法"
    - 项目选择器, "选择器语法"
    - ip区间
      - CIDR
      - ip中的例外列表
  - ...

QINGCLOUD

查看网络连通性

- 使用 kubectl describe networkpo
  看
- 在被禁止和被允许的 Pod 上测试
- 测试外部网络
  - 入口：是否允许外部 LB 访问
  - 出口：是否允许访问外部网络
- 测试其他 Namespace 和 Port

```
  # kubectl describe networkpolicies web-allow-all-ns-
monitoring
Name:         web-allow-all-ns-monitoring
Namespace:    default
Created on:   2019-08-20 13:43:21 +0800 CST
Labels:       <none>
Annotations:  kubectl.kubernetes.io/last-applied-
configuration:
                {"apiVersion":...
Spec:
  PodSelector:     app=web
  Allowing ingress traffic:
    To Port: <any> (traffic allowed to all ports)
    From:
      NamespaceSelector: team=operations
      PodSelector: type=monitoring
  Allowing egress traffic:
    <none> (Selected pods are isolated for egress
connectivity)
  Policy Types: Ingress
```

QINGCLOUD

NetworkPolicy on KubeSphere

- 2.1 将会提供 WorkspaceNetworkPolicy / NamespaceNetworkPolicy
- 3.0 将会提供 UI 界面，可以快速创建和审查你创建的 NetworkPolicy
- 结合 Calico：
  - 提供策略排序/优先级
  - 拒绝规则
  - 日志
  - 更多的协议支持（TCP, UDP, ICMP, SCTP, UDPlite, ICMPv6, 协议号 1-255）
  - HTTP 支持（启用 istio 时）
  - 更多的匹配规则（notSelector/notPorts/contains/"starts with"/"ends with"）

QINGCLOUD

参考样例

- https://github.com/ahmetb/kubernetes-network-policy-recipes
- https://medium.com/@reuvenharrison/an-introduction-to-kubernetes-network-policies-for-security-people-ba92dd4c809d

QINGCLOUD

Thanks