链滴

CKA 每日一题第 10 天 | （网络隔离）允许 A 访问 B，不允许 C 访问 B

作者：liabio

原文链接：https://ld246.com/article/1583916752609

来源网站：链滴

昨日考题

部署三个deployment应用(A,B,C),允许A访问B应用,但是不允许C访问B应用。

- Deployment的名称为cka-1128-01，cka-1128-02，cka-1128-03
- Network Policy的名称为cka-1128-np

注意：将所用命令、创建的deployment以及network policy完整yaml、以及证明A可以访问B应；C不允许访问B应用。可分多次评论。

昨日答案

第一个Deploy文件cka-1128-01.yaml，使用radial/busyboxplus镜像是因为busybox里没有curl命。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: cka-1128-01
spec:
  selector:
    matchLabels:
      app: cka-1128-01
  template:
    metadata:
      labels:
        app: cka-1128-01
    spec:
      containers:
        - name: cka-1128-01
          image: radial/busyboxplus
          command: ['sh', '-c', 'sleep 1000']
          imagePullPolicy: IfNotPresent
```

cka-1128-02.yaml：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: cka-1128-02
spec:
  selector:
    matchLabels:
      app: cka-1128-02
  template:
    metadata:
      labels:
        app: cka-1128-02
    spec:
      containers:
        - name: cka-1128-02
          image: radial/busyboxplus
          command: ['sh', '-c', 'sleep 1000']
```

```
        imagePullPolicy: IfNotPresent
```

cka-1128-03.yaml：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: cka-1128-03
spec:
  selector:
    matchLabels:
      app: cka-1128-03
  template:
    metadata:
      labels:
        app: cka-1128-03
    spec:
      containers:
        - name: cka-1128-03
          image: radial/busyboxplus
          command: ['sh', '-c', 'sleep 1000']
          imagePullPolicy: IfNotPresent
```

可以看到A、C都可以访问B：

```
[root@liabio cka]# kubectl get pod -owide | grep cka
cka-1128-01-7b8b8cb79-mll6d       1/1     Running   0          3m5s   192.168.155.124   liabio
none>           <none>
cka-1128-02-69dd65bdb7-mfq26      1/1     Running   0          3m8s   192.168.155.117   liabio
<none>          <none>
cka-1128-03-66f8f69ff-64q75       1/1     Running   0          3m3s   192.168.155.116   liabio   <
one>            <none>
[root@liabio cka]# kubectl exec -ti cka-1128-01-7b8b8cb79-mll6d  -- ping 192.168.155.117
PING 192.168.155.117 (192.168.155.117): 56 data bytes
64 bytes from 192.168.155.117: seq=0 ttl=63 time=0.146 ms
64 bytes from 192.168.155.117: seq=1 ttl=63 time=0.095 ms
[root@liabio cka]# kubectl exec -ti cka-1128-03-66f8f69ff-64q75  -- ping 192.168.155.117
PING 192.168.155.117 (192.168.155.117): 56 data bytes
64 bytes from 192.168.155.117: seq=0 ttl=63 time=0.209 ms
64 bytes from 192.168.155.117: seq=1 ttl=63 time=0.112 ms
```

新建cka-1128-np.yaml，kubectl apply -f cka-1128-np.yaml创建Network Policy，spec.podSelector.matchLabels选择B管理的Pod；ingress.from.podSelector.matchLabels指定只给来自A的流量白名单。

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: cka-1128-np
spec:
  podSelector:
    matchLabels:
      app: cka-1128-02
  ingress:
```

```
  - from:
    - podSelector:
        matchLabels:
          app: cka-1128-01
```

验证发现A可以ping通B，C不能ping通B。

```
[root@liabio cka]# kubectl apply -f cka-1128-np.yaml
networkpolicy.networking.k8s.io/cka-1128-np created
[root@liabio cka]# kubectl get networkpolicies.
NAME          POD-SELECTOR      AGE
cka-1128-np   app=cka-1128-02   13s
[root@liabio cka]#
[root@liabio cka]# kubectl get pod -owide | grep cka
cka-1128-01-7b8b8cb79-mll6d       1/1     Running   1          24m    192.168.155.124   liabio
none>           <none>
cka-1128-02-69dd65bdb7-mfq26      1/1     Running   1          24m    192.168.155.117   liabio
<none>          <none>
cka-1128-03-66f8f69ff-64q75       1/1     Running   1          24m    192.168.155.116   liabio   <
one>            <none>
[root@liabio cka]# kubectl exec -ti cka-1128-01-7b8b8cb79-mll6d  -- ping 192.168.155.117
PING 192.168.155.117 (192.168.155.117): 56 data bytes
64 bytes from 192.168.155.117: seq=0 ttl=63 time=0.213 ms

[root@liabio cka]# kubectl exec -ti cka-1128-03-66f8f69ff-64q75  -- ping 192.168.155.117
PING 192.168.155.117 (192.168.155.117): 56 data bytes
......
```

昨日解析

本题的关键点就是考察k8s网络策略NetworkPolicy，这块知识点面试中也经常会被问到。以下摘抄官方文档：

概念

网络策略（NetworkPolicy）是一种关于pod间及pod与其他网络端点间所允许的通信规则的规范。

NetworkPolicy资源使用标签选择pod，并定义选定pod所允许的通信规则。网络策略通过网络插件实现，所以用户必须使用支持 NetworkPolicy 的网络解决方案，

网络策略概念与介绍官方文档：

https://kubernetes.io/docs/concepts/services-networking/network-policies/#the-networkpolicy-resource

网络策略实践官方文档，本题目完全可以参考该文档完成。

https://kubernetes.io/docs/tasks/administer-cluster/declare-network-policy/

前提

网络策略通过网络插件来实现，所以用户必须使用支持 NetworkPolicy 的网络解决方案 - 简单地创资源对象，而没有控制器来使它生效的话，是没有任何作用的。

支持的网络方案

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

k8s官方文档说明：

https://kubernetes.io/docs/tasks/administer-cluster/network-policy-provider/

隔离和非隔离的Pod

默认情况下，Pod是非隔离的，它们接受任何来源的流量。

Pod可以通过相关的网络策略进行隔离。一旦命名空间中有NetworkPolicy选择了特定的Pod，该Po会拒绝网络策略所不允许的连接。 (命名空间下其他未被网络策略所选择的Pod会继续接收所有的流量)

NetworkPolicy 资源

下面是一个 NetworkPolicy 的示例:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - from:
    - ipBlock:
        cidr: 172.17.0.0/16
        except:
        - 172.17.1.0/24
    - namespaceSelector:
        matchLabels:
          project: myproject
    - podSelector:
        matchLabels:
          role: frontend
    ports:
    - protocol: TCP
      port: 6379
  egress:
  - to:
    - ipBlock:
        cidr: 10.0.0.0/24
    ports:
    - protocol: TCP
      port: 5978
```

除非选择支持NetworkPolicy 的网络解决方案，否则将上述示例发送到APIServer没有任何效果。

spec: NetworkPolicy spec 中包含了在一个命名空间中定义特定网络策略所需的所有信息

podSelector: 每个 NetworkPolicy 都包括一个 podSelector ，它对该策略所应用的一组Pod进行择。因为 NetworkPolicy 目前只支持定义 ingress 规则，这里的 podSelector 本质上是为该策略定义“目标pod” 。示例中的策略选择带有 “role=db” 标签的pod。空的 podSelector 选择命名空间的所有pod。

policyTypes：每个NetworkPolicy都包含一个policyTypes列表，其中可能包含Ingress，Egress或者。 policyTypes字段指示给定的策略是否适用于到选定Pod的入站流量，来自选定Pod的出站流量或两者都适用。 如果在NetworkPolicy上未指定任何policyType，则默认情况下将始终设置Ingress并且如果NetworkPolicy具有任何出口规则，则将设置Egress。

ingress: 每个 NetworkPolicy 包含一个 ingress 规则的白名单列表。（其中的）规则允许同时匹配 fr m 和 ports 部分的流量。示例策略中包含一条简单的规则： 它匹配一个单一的port，来自两个来源的一个， 第一个通过 namespaceSelector 指定，第二个通过 podSelector 指定。

egress: 每个 NetworkPolicy 包含一个 egress 规则的白名单列表。每个规则都允许匹配 to 和 port 分的流量。该示例策略包含一条规则，该规则将单个端口上的流量匹配到 10.0.0.0/24 中的任何目的。

所以，示例网络策略:

隔离 “default” 命名空间下 “role=db” 的pod (如果它们不是已经被隔离的话)。

允许从 “default” 命名空间下带有 “role=frontend” 标签的pod到 “default” 命名空间下的po 的6379 TCP端口的连接。

- “default” 名称空间中，带有标签为 “role=frontend” 的任何Pod；
- namespaces中带有标签“project=myproject” 的任何pod；
- IP 地址范围为 172.17.0.0–172.17.0.255 和

172.17.2.0–172.17.255.255（即，除了 172.17.1.0/24 之外的所有 172.17.0.0/16）

允许从带有 “project=myproject” 标签的命名空间下的任何 pod 到 “default” 命名空间下的 po 的6379 TCP端口的连接。

选择器 to 和 from 的行为

可以在 ingress from 部分或 egress to 部分中指定四种选择器：

podSelector: 这将在与 NetworkPolicy 相同的名称空间中选择特定的 Pod，应将其允许作为入口源出口目的地。

namespaceSelector: 这将选择特定的名称空间，应将所有 Pod 用作其输入源或输出目的地。

namespaceSelector 和 podSelector: 一个指定 namespaceSelector 和 podSelector 的 to/from 条目选择特定命名空间中的特定 Pod。注意使用正确的YAML语法；这项策略：

```
...
  ingress:
  - from:
    - namespaceSelector:
```

```
      matchLabels:
        user: alice
    podSelector:
      matchLabels:
        role: client
  ...
```

在 from 数组中仅包含一个元素，只允许来自标有 role = client 的 Pod 且该 Pod 所在的名称空间中有user=alice的连接。这项策略：

```
  ...
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          user: alice
    - podSelector:
        matchLabels:
          role: client
  ...
```

在 from 数组中包含两个元素，允许来自本地命名空间中标有 role = client 的 Pod 的连接，或来自何名称空间中标有user = alice的任何Pod的连接。

ipBlock: 这将选择特定的 IP CIDR 范围以用作入口源或出口目的地。 这些应该是集群外部 IP，因为od IP 存在时间短暂的且随机产生。

集群的入口和出口机制通常需要重写数据包的源 IP 或目标 IP。在发生这种情况的情况下，不确定在etworkPolicy 处理之前还是之后发生，并且对于网络插件，云提供商，Service 实现等的不同组合，行为可能会有所不同。

在进入的情况下，这意味着在某些情况下，您可以根据实际的原始源 IP 过滤传入的数据包，而在其情况下，NetworkPolicy 所作用的 源IP 则可能是 LoadBalancer 或 Pod的节点等。

对于出口，这意味着从 Pod 到被重写为集群外部 IP 的 Service IP 的连接可能会或可能不会受到基于 iBlock 的策略的约束。

默认策略

默认情况下，如果名称空间中不存在任何策略，则所有进出该名称空间中的Pod的流量都被允许。以示例使您可以更改该名称空间中的默认行为。

默认拒绝所有入口流量

您可以通过创建选择所有容器但不允许任何进入这些容器的入口流量的 NetworkPolicy 来为名称空创建 “default” 隔离策略。

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

```
  policyTypes:
  - Ingress
```

这样可以确保即使容器没有选择其他任何 NetworkPolicy，也仍然可以被隔离。此策略不会更改默认出口隔离行为。

默认允许所有入口流量

如果要允许所有流量进入某个命名空间中的所有 Pod（即使添加了导致某些 Pod 被视为“隔离”的略），则可以创建一个策略来明确允许该命名空间中的所有流量。

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

默认拒绝所有出口流量

您可以通过创建选择所有容器但不允许来自这些容器的任何出口流量的 NetworkPolicy 来为名称空创建 “default” egress 隔离策略。

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

这样可以确保即使没有被其他任何 NetworkPolicy 选择的 Pod 也不会被允许流出流量。此策略不会改默认的 ingress 隔离行为。

默认允许所有出口流量

如果要允许来自命名空间中所有 Pod 的所有流量（即使添加了导致某些 Pod 被视为“隔离”的策略，则可以创建一个策略，该策略明确允许该命名空间中的所有出口流量。

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

```
  - Egress
```

默认拒绝所有入口和所有出口流量

您可以为名称空间创建 “default” 策略，以通过在该名称空间中创建以下 NetworkPolicy 来阻止有入站和出站流量。

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

这样可以确保即使没有被其他任何 NetworkPolicy 选择的 Pod 也不会被允许进入或流出流量。

今日考题

创建一个Role(只有cka namespace下pods的所有操作权限)和RoleBinding(使用serviceaccount认鉴权),使用对应serviceaccount作为认证信息对cka namespace下的pod进行操作以及对default namspace下的pods进行操作。

– Role和RoleBinding的名称的名称为cka-1202-role、cka-1202-rb

注意：请附所用命令、创建的Role、RoleBinding以及serviceaccount的完整yaml，可分多次评论。

作者简介

作者：小碗汤，一位热爱、认真写作的小伙，目前维护原创公众号：『我的小碗汤』，专注于写linux golang、docker、kubernetes等知识等提升硬实力的文章，期待你的关注。转载说明：务必注明来（注明：来源于公众号：我的小碗汤， 作者：小碗汤）