OpenShift Container Platform 4.19

网络安全性

在 OpenShift Container Platform 中保护网络流量并强制实施网络策略

Last Updated: 2025-12-11

OpenShift Container Platform 4.19 网络安全性

在 OpenShift Container Platform 中保护网络流量并强制实施网络策略

Legal Notice

Abstract

本文档论述了如何在 OpenShift Container Platform 中实现和管理网络安全功能，如网络策略和出口防火墙。

Table of Contents

第 1 章 了解网络策略 API

网络策略使用集群范围的网络策略 API 定义。通过在这些不同级别上定义网络策略，您可以为集群创建复杂的网络安全配置，包括完整的多租户隔离。

1.1. 网络策略及其范围

集群范围的网络策略

集群和网络管理员可以使用 AdminNetworkPolicy 在集群级别上定义网络策略。AdminNetworkPolicy 功能由两个 API 组成： **AdminNetworkPolicy** API 和 **BaselineAdminNetworkPolicy** API。这些 API 用于设置可应用到整个集群的规则，或委派给命名空间范围的 **NetworkPolicy**。

当设置为 "Allow" 或 "Deny" 时，使用 **AdminNetworkPolicy** API 定义的策略优先于所有其他策略类型。但是，管理员也可以使用"Pass"将给定策略负责委派给命名空间范围的 **NetworkPolicy**，以允许应用程序开发人员和命名空间租户控制其项目网络安全性的特定方面。

使用 **BaselineAdminNetworkPolicy** API 定义的策略仅在没有其他网络策略覆盖它们时才适用。当您使用 **AdminNetworkPolicy** API 将网络策略方面委托给命名空间范围的 **NetworkPolicy** 时，您还应在 **BaselineAdminNetworkPolicy** 中定义合理的最小限制。这样可确保在命名空间 **NetworkPolicy** 没有足够保护时，集群级别的网络安全性基准级别。

命名空间范围的网络策略

应用程序开发人员和命名空间租户可以使用 **NetworkPolicy** API 为特定命名空间定义网络策略规则。一个命名空间的 **NetworkPolicy** 中的规则优先于使用 BaselineAdminNetworkPolicy API 配置的集群范围规则，或用于集群范围 **AdminNetworkPolicy** API 中委派或"passed"的集群范围规则。

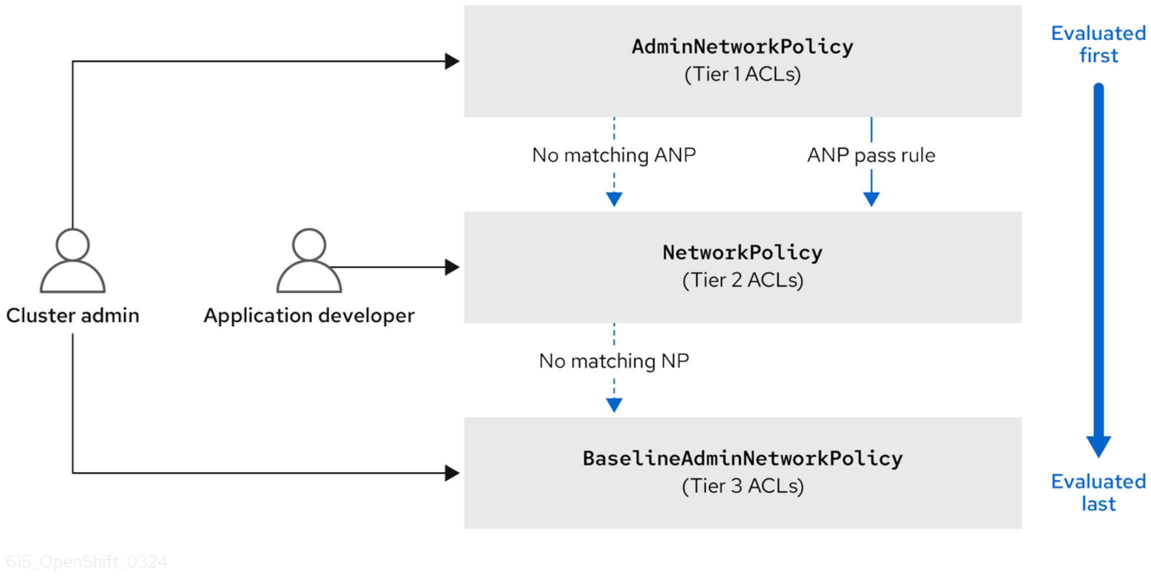
1.2. 如何评估和应用网络策略

当建立网络连接时，网络供应商（默认为 OVN-Kubernetes）会根据网络策略规则检查连接详情，以确定如何处理连接。

OVN-Kubernetes 按以下顺序评估针对网络策略对象的连接：

1. 在 AdminNetworkPolicy 层检查是否有匹配项。
    a. 如果连接与 **Allow** 或 **Deny** 规则匹配，请按照该规则和停止评估。
    b. 如果连接与 **Pass** 规则匹配，请移到 NetworkPolicy 层。
2. 在 NetworkPolicy 层检查是否有匹配项。
    a. 如果连接与规则匹配，请按照该规则进行操作并停止评估。
    b. 如果没有找到匹配项，移到 BaselineAdminNetworkPolicy 层。
3. 遵循 BaselineAdminNetworkPolicy 层中的匹配规则。

图 1.1. OVN-Kubernetes 评估网络策略



1.3. ADMINNETWORKPOLICY 和 NETWORKPOLICY 自定义资源之间的主要区别

下表解释了集群范围的 **AdminNetworkPolicy** API 和命名空间范围 **NetworkPolicy** API 之间的主要区别。

策略元素	AdminNetworkPolicy	NetworkPolicy
适用的用户	集群管理员	命名空间所有者
影响范围	Cluster	Namespace
丢弃流量	当显式 **Deny** 操作设置为一个规则时支持。	在策略创建时通过隐式 **Deny** 隔离时支持。
委派流量	**Pass** 操作设置为一个规则时被支持。	不适用。
允许流量	显式 **Allow** 操作设置一个规则时被支持。	所有规则的默认操作都是 allow。
策略中的规则优先级	取决于它们出现在 ANP 中的顺序。位置更高的规则有更高的优先级。	规则的效果是叠加的。
策略优先级	在 ANP 中，**priority** 字段用于设置评估的顺序。策略优先级号越低，优先级越高。	在策略之间没有策略排序。

策略元素	AdminNetworkPolicy	NetworkPolicy
功能优先级	首先通过 1 层 ACL 和 BANP 评估，最后通过第 3 层 ACL 评估。	在 ANP 之后，BANP 之前强制实施，它们会在 ACL 层 2 中进行评估。
匹配 pod 选择	在命名空间之间可应用不同的规则。	可以在单一命名空间中的 pod 之间应用不同的规则。
集群出口流量	通过**节点**和**网络** 对等点支持	通过 **ipBlock** 字段以及接受的 CIDR 语法支持。
集群入口流量	不支持。	不支持。
完全限定域名 (FQDN) 对等支持	不支持。	不支持。
命名空间选择器	支持通过使用 **namespaces.matchLabels** 字段进行命名空间的高级选择。	支持使用 **namespaceSelector** 字段支持基于标签的命名空间选择。

第 2 章 管理网络策略

2.1. OVN-KUBERNETES ADMINNETWORKPOLICY

2.1.1. AdminNetworkPolicy

AdminNetworkPolicy (ANP)是一个集群范围的自定义资源定义(CRD)。作为 OpenShift Container Platform 管理员，您可以在创建命名空间前通过创建网络策略来使用 ANP 来保护网络。另外，您可以在集群范围的级别上创建网络策略，该级别不可由 **NetworkPolicy** 对象覆盖。

AdminNetworkPolicy 和 **NetworkPolicy** 对象之间的关键区别在于，供管理员使用，是集群范围，而后者则用于租户所有者，并且是命名空间范围。

ANP 允许管理员指定以下内容：

- 确定其评估顺序的 **priority** 值。数值越低，优先级越高。
- 由应用策略的一组命名空间或命名空间组成的一组 pod。
- 要应用到 **subject** 的所有入口流量的入站规则列表。
- 用于来自 **subject** 的所有出口流量的出口规则列表。

2.1.1.1. AdminNetworkPolicy 示例

例 2.1. ANP 的 YAML 文件示例

```
apiVersion: policy.networking.k8s.io/v1alpha1
kind: AdminNetworkPolicy
metadata:
  name: sample-anp-deny-pass-rules 1
spec:
  priority: 50 2
  subject:
    namespaces:
      matchLabels:
          kubernetes.io/metadata.name: example.name 3
  ingress: 4
  - name: "deny-all-ingress-tenant-1" 5
    action: "Deny"
    from:
    - pods:
        namespaceSelector:
          matchLabels:
            custom-anp: tenant-1
        podSelector:
          matchLabels:
            custom-anp: tenant-1 6
  egress: 7
  - name: "pass-all-egress-to-tenant-1"
    action: "Pass"
    to:
    - pods:
```

```
      namespaceSelector:
        matchLabels:
          custom-anp: tenant-1
      podSelector:
        matchLabels:
          custom-anp: tenant-1
```

1. 为您的 ANP 指定一个名称。
2. **spec.priority** 字段支持在一个集群中最大 100 ANP（范围为 **0-99**）。数越低，优先级越高，因为范围是从按最低到最高值的顺序读取的。因为当以同一优先级创建 ANP 时，无法保证哪些策略会被优先使用，所以请使用不同的优先级来设置 ANPs。
3. 指定要应用 ANP 资源的命名空间。
4. ANP 具有入口和出口规则。**spec.ingress** 字段的 ANP 规则接受 **Pass**,**Deny**, **action** 字段接受的值为 **Allow**。
5. 为 **ingress.name** 指定一个名称。
6. 指定 **podSelector.matchLabels**，以选择 **namespaceSelector.matchLabels** 作为入口对等选择的命名空间中的 pod。
7. ANP 同时具有入口和出口规则。**spec.egress** 字段的 ANP 规则接受 **Pass**,**Deny**, **action** 字段接受的值为 **Allow**。

其他资源

- Network Policy API 工作组

2.1.1.2. 规则的 AdminNetworkPolicy 操作

作为管理员，您可以将您的 **AdminNetworkPolicy** 规则的 **action** 字段设置为 **Allow**,**Deny**, 或 **Pass**。由于 OVN-Kubernetes 使用分层 ACL 来评估网络流量规则，因此 3NP 允许您设置非常强大的策略规则，它们只能被管理员修改、删除规则，或通过设置更高优先级规则来覆盖它们。

2.1.1.2.1. AdminNetworkPolicy Allow 示例

在优先级 9 中定义的以下 ANP 可确保允许从 **monitoring** 命名空间到集群中的任何租户（所有其他命名空间）的所有入口流量。

例 2.2. 强 Allow ANP 的 YAML 文件示例

```
apiVersion: policy.networking.k8s.io/v1alpha1
kind: AdminNetworkPolicy
metadata:
  name: allow-monitoring
spec:
  priority: 9
  subject:
    namespaces: {} # Use the empty selector with caution because it also selects OpenShift
namespaces as well.
  ingress:
```

```yaml
  - name: "allow-ingress-from-monitoring"
    action: "Allow"
    from:
    - namespaces:
        matchLabels:
          kubernetes.io/metadata.name: monitoring
# ...
```

这是强的 **Allow** ANP 的示例，因为它不可以被涉及的所有方覆盖。租户都不会阻止自己被使用 **NetworkPolicy** 对象监控，监控租户也不知道它可以或无法监控的内容。

2.1.1.2.2. AdminNetworkPolicy 拒绝示例

在优先级 5 中定义的以下 ANP 可确保 **monitoring** 命名空间中的所有入口流量都被阻止到受限租户（具有标签 **security: restricted**的命名空间）。

例 2.3. 强 Deny ANP 的 YAML 文件示例

```yaml
apiVersion: policy.networking.k8s.io/v1alpha1
kind: AdminNetworkPolicy
metadata:
  name: block-monitoring
spec:
  priority: 5
  subject:
    namespaces:
      matchLabels:
        security: restricted
  ingress:
  - name: "deny-ingress-from-monitoring"
    action: "Deny"
    from:
    - namespaces:
        matchLabels:
          kubernetes.io/metadata.name: monitoring
# ...
```

这是一个强大的 **Deny** ANP，这是所有涉及的方都无法覆盖的。受限租户所有者无法授权自己允许监控流量，基础架构监控服务无法从这些敏感命名空间中提取任何内容。

与强的 **Allow** 示例结合使用时，**block-monitoring** ANP 具有较低优先级的值，赋予其优先级更高的优先级，这样可确保不会监控受限租户。

2.1.1.2.3. AdminNetworkPolicy Pass 示例

在优先级 7 定义的以下 ANP 可确保所有从 **monitoring** 命名空间到内部基础架构租户（具有标签 **security: internal**）的入口流量都将传递到 ACL 的层 2，并由命名空间的 **NetworkPolicy** 对象评估。

例 2.4. 强 Pass ANP 的 YAML 文件示例

```yaml
apiVersion: policy.networking.k8s.io/v1alpha1
```

```yaml
kind: AdminNetworkPolicy
metadata:
  name: pass-monitoring
spec:
  priority: 7
  subject:
    namespaces:
      matchLabels:
        security: internal
  ingress:
  - name: "pass-ingress-from-monitoring"
    action: "Pass"
    from:
    - namespaces:
        matchLabels:
          kubernetes.io/metadata.name: monitoring
# ...
```

这个示例是一个强大的 **Pass** 操作 ANP，因为它将决策委派给租户所有者定义的 **NetworkPolicy** 对象。如果基础架构监控服务应使用命名空间范围 **NetworkPolicy** 对象提取其指标，则此 **pass-monitoring** ANP 允许在安全级别 **internal** 分组的所有租户所有者。

2.2. OVN-KUBERNETES BASELINEADMINNETWORKPOLICY

2.2.1. BaselineAdminNetworkPolicy

BaselineAdminNetworkPolicy (BANP)是一个集群范围的自定义资源定义(CRD)。作为 OpenShift Container Platform 管理员，您可以使用 BANP 来设置并强制实施可选的基准网络策略规则，这些规则被用户使用 **NetworkPolicy** 对象（如果需要的话）覆盖。BANP 的规则操作是 **allow** 或 **deny**。

BaselineAdminNetworkPolicy 资源是一个集群单例对象，当传递的流量策略与集群中的任何 **NetworkPolicy** 对象不匹配时，可用作 guardrail 策略。BANP 也可以用作默认安全模型，该模型默认阻止集群内流量，用户需要使用 **NetworkPolicy** 对象来允许已知的流量。在创建 BANP 资源时，必须使用 **default** 作为名称。

管理员可通过 BANP 指定：

- 由一组命名空间或命名空间的 **subject**。
- 要应用到 **subject** 的所有入口流量的入站规则列表。
- 用于来自 **subject** 的所有出口流量的出口规则列表。

2.2.1.1. BaselineAdminNetworkPolicy 示例

例 2.5. BANP 的 YAML 文件示例

```yaml
apiVersion: policy.networking.k8s.io/v1alpha1
kind: BaselineAdminNetworkPolicy
metadata:
  name: default (1)
spec:
```

```
subject:
  namespaces:
    matchLabels:
      kubernetes.io/metadata.name: example.name 2
ingress: 3
- name: "deny-all-ingress-from-tenant-1" 4
  action: "Deny"
  from:
  - pods:
      namespaceSelector:
        matchLabels:
          custom-banp: tenant-1 5
      podSelector:
        matchLabels:
          custom-banp: tenant-1 6
egress:
- name: "allow-all-egress-to-tenant-1"
  action: "Allow"
  to:
  - pods:
      namespaceSelector:
        matchLabels:
          custom-banp: tenant-1
      podSelector:
        matchLabels:
          custom-banp: tenant-1
```

1. 策略名称必须是 **default**，因为 BANP 是一个单例对象。
2. 指定要将 ANP 应用到的命名空间。
3. BANP 具有入口和出口规则。**spec.ingress** 和 **spec.egress** 字段的 BANP 规则接受 **Deny**，**action** 字段接受的值为 **Allow**。
4. 为 **ingress.name** 指定名称
5. 指定要从中选择 pod 以应用 BANP 资源的命名空间。
6. 指定 **podSelector.matchLabels** 名称，以应用 BANP 资源。

2.2.1.2. BaselineAdminNetworkPolicy 拒绝示例

以下 BANP 单例确保管理员为 **internal** 安全级别进入租户的所有入口监控流量设置了默认的拒绝策略。与 "AdminNetworkPolicy Pass example" 组合时，这个 deny 策略充当 ANP **pass-monitoring** 策略传递的所有入口流量的保护策略。

例 2.6. guardrail Deny 规则的 YAML 文件示例

```
apiVersion: policy.networking.k8s.io/v1alpha1
kind: BaselineAdminNetworkPolicy
metadata:
  name: default
```

```
spec:
  subject:
    namespaces:
      matchLabels:
        security: internal
  ingress:
  - name: "deny-ingress-from-monitoring"
    action: "Deny"
    from:
    - namespaces:
        matchLabels:
          kubernetes.io/metadata.name: monitoring
# ...
```

您可以将带有 **action** 字段的值为 **Pass** 的 **AdminNetworkPolicy** 资源与 **BaselineAdminNetworkPolicy** 资源结合使用来创建多租户策略。此多租户策略允许一个租户在应用上收集监控数据，同时不从第二个租户收集数据。

作为管理员，如果您同时应用了 "AdminNetworkPolicy **Pass** action example" 和 "BaselineAdminNetwork Policy **Deny** example"，则租户将保留创建在 BANP 之前评估的 **NetworkPolicy** 资源。

例如，租户 1 可以设置以下 **NetworkPolicy** 资源来监控入口流量：

例 2.7. NetworkPolicy 示例

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-monitoring
  namespace: tenant 1
spec:
  podSelector:
  policyTypes:
    - Ingress
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          kubernetes.io/metadata.name: monitoring
# ...
```

在这种情况下，Tenant 1 会在 "AdminNetworkPolicy **Pass** action example" 之后，"BaselineAdminNetwork Policy **Deny** example" 之前被评估，它将拒绝所有进入 **安全**级别 **internal** 的入口监控流量。随着租户 1 的 **NetworkPolicy** 对象就位，它们将能够在其应用程序中收集数据。但是，租户 2 没有任何 **NetworkPolicy** 对象，将无法收集数据。作为管理员，您没有默认监控内部租户，而是创建了 BANP，它允许租户使用 **NetworkPolicy** 对象覆盖 BANP 的默认行为。

2.3. 监控 ANP 和 BANP

AdminNetworkPolicy 和 **BaselineAdminNetworkPolicy** 资源具有可用于监控和管理您的策略的指标。有关指标的详情，请查看下表。

2.3.1. AdminNetworkPolicy 指标

Name	描述	解释
ovnkube_controller_admin_network_policies	Not applicable	集群中的 **AdminNetworkPolicy** 资源总数。
ovnkube_controller_baseline_admin_network_policies	Not applicable	集群中的 **BaselineAdminNetworkPolicy** 资源总数。该值应该是 0 或 1。
ovnkube_controller_admin_network_policies_rules	• **direction**: 指定 **Ingress** 或 **Egress**。 • **action**: 指定 **Pass**, **Allow**, 或 **Deny**。	集群中所有 ANP 策略的规则总数，按照 **direction** 和 **action** 分组。
ovnkube_controller_baseline_admin_network_policies_rules	• **direction**: 指定 **Ingress** 或 **Egress**。 • **action** ：指定 **Allow** 或 **Deny**。	集群中所有 BANP 策略的规则总数，按照 **direction** 和 **action** 分组。
ovnkube_controller_admin_network_policies_db_objects	**table_name** ：指定 **ACL** 或 **Address_Set**	集群中所有 ANP 创建的 OVN 北向数据库(nbdb)对象的总数，按照 **table_name** 分组。
ovnkube_controller_baseline_admin_network_policies_db_objects	**table_name** ：指定 **ACL** 或 **Address_Set**	集群中所有 BANP 创建的 OVN 北向数据库(nbdb)对象的总数，按照 **table_name** 分组。

2.4. ADMINNETWORKPOLICY 的出口节点和网络对等点

本节介绍 **节点 和网络** 对等点。管理员可以使用本节中的示例来设计 **AdminNetworkPolicy** 和 **BaselineAdminNetworkPolicy**，以控制其集群中的北向流量。

2.4.1. AdminNetworkPolicy 和 BaselineAdminNetworkPolicy 的北向流量控制

除了支持 east-west 流量控制外，ANP 和 BANP 还允许管理员控制其北向流量，使集群或流量离开集群或流量到集群中的其他节点。最终用户可以执行以下操作：

- 使用 **节点** 出口对等点实现对集群节点的出口流量控制
- 使用**节点**或**网络** 出口对等对 Kubernetes API 服务器实施出口流量控制
- 使用**网络**对等点对集群外的外部目的地实施出口流量控制

> **注意**
>
> 对于 ANP 和 BANP，只能为出口规则指定**节点**和**网络**对等点。

2.4.1.1. 使用节点 peer 控制到集群节点的出口流量

使用 **节点** 对等管理员可以控制从 pod 到集群中节点的出口流量。这样做的好处是，您不必在向集群添加或删除节点时更改策略。

在以下示例中，通过使用节点选择器，允许任何带有**restricted**, **confidential**, 或 **internal** 级别安全的命名空间发送的、端口 **6443** 上的到 Kubernetes API 服务器的出口流量。它还拒绝来自带有 **restricted**, **confidential**, or **internal** 安全级别的任何命名空间的、到您的集群中的所有 worker 节点的流量。

例 2.8. 使用 nodes 对等的 ANP Allow egress 示例

```
apiVersion: policy.networking.k8s.io/v1alpha1
kind: AdminNetworkPolicy
metadata:
  name: egress-security-allow
spec:
  egress:
  - action: Deny
    to:
    - nodes:
        matchExpressions:
        - key: node-role.kubernetes.io/worker
          operator: Exists
  - action: Allow
    name: allow-to-kubernetes-api-server-and-engr-dept-pods
    ports:
    - portNumber:
        port: 6443
        protocol: TCP
    to:
    - nodes: 1
        matchExpressions:
        - key: node-role.kubernetes.io/control-plane
          operator: Exists
    - pods: 2
        namespaceSelector:
          matchLabels:
            dept: engr
        podSelector: {}
  priority: 55
  subject: 3
    namespaces:
      matchExpressions:
      - key: security 4
        operator: In
        values:
        - restricted
        - confidential
        - internal
```

1. 使用 **matchExpressions** 字段指定集群中的节点或一组节点。
2. 指定使用 **dept: engr** 标记的所有 pod。
3. 指定 ANP 的主题，其中包含任何与网络策略使用的标签匹配的命名空间。示例匹配任何带有 **security** 的级别为 **restricted**, **confidential**, 或 **internal** 级别的命名空间。
4. 为 **matchExpressions** 字段指定键/值对。

2.4.1.2. 使用网络对等控制到外部目的地的出口流量

集群管理员可以使用**网络**对等中的 CIDR 范围，并应用一个策略来控制离开 pod 的出口流量，并进入通过 **network** 字段指定的 CIDR 范围内配置的 IP 地址的目标。

以下示例使用**网络**对等，并组合了 ANP 和 BANP 策略来限制出口流量。

重要

请谨慎使用 **namespace** 字段中的空选择器({})。使用空选择器时，它还选择 OpenShift 命名空间。

如果您在 ANP 或 BANP **Deny** 规则中使用 **0.0.0.0/0** 的值，您必须在将 **Deny** 设置为 **0.0.0.0/0** 前将更高的优先级 ANP **Allow** 规则设置为所需的目的地。

例 2.9. 使用网络对等点的 ANP 和 BANP 示例

```
apiVersion: policy.networking.k8s.io/v1alpha1
kind: AdminNetworkPolicy
metadata:
  name: network-as-egress-peer
spec:
  priority: 70
  subject:
    namespaces: {} # Use the empty selector with caution because it also selects OpenShift
namespaces as well.
  egress:
  - name: "deny-egress-to-external-dns-servers"
    action: "Deny"
    to:
    - networks: (1)
      - 8.8.8.8/32
      - 8.8.4.4/32
      - 208.67.222.222/32
    ports:
      - portNumber:
          protocol: UDP
          port: 53
  - name: "allow-all-egress-to-intranet"
    action: "Allow"
    to:
    - networks: (2)
      - 89.246.180.0/22
```

```
      - 60.45.72.0/22
  - name: "allow-all-intra-cluster-traffic"
    action: "Allow"
    to:
    - namespaces: {} # Use the empty selector with caution because it also selects OpenShift namespaces as well.
  - name: "pass-all-egress-to-internet"
    action: "Pass"
    to:
    - networks:
      - 0.0.0.0/0 (3)
---
apiVersion: policy.networking.k8s.io/v1alpha1
kind: BaselineAdminNetworkPolicy
metadata:
  name: default
spec:
  subject:
    namespaces: {} # Use the empty selector with caution because it also selects OpenShift namespaces as well.
  egress:
  - name: "deny-all-egress-to-internet"
    action: "Deny"
    to:
    - networks:
      - 0.0.0.0/0 (4)
---
```

1. 使用**网络**指定集群外的 CIDR 网络范围。
2. 指定来自资源的集群内流量的 CIDR 范围。
3. 4. 通过将 **networks** 值设置为 **0.0.0.0/0** 来指定到任何项的 **Deny** egress。在将 **Deny** 设置为 **0.0.0.0/0** 之前，请确保具有较高的优先级 **Allow** 规则到所需的目的地，因为这将拒绝包括到 Kubernetes API 和 DNS 服务器的所有流量。

使用**网络**对等来整合 **network-as-egress-peer** ANP 和 **默认**的 BANP 来强制以下 egress 策略：

- 所有 pod 都无法通过列出的 IP 地址与外部 DNS 服务器进行通信。
- 所有 pod 都可以与公司的其他内部网通信。
- 所有 pod 都可以与其他 pod、节点和服务通信。
- 所有 pod 都无法与互联网通信。将最后一个 ANP **Pass** 规则与强大的 BANP **Deny** 规则合并会创建一个保护策略来保护集群中的流量。

2.4.1.3. 一起使用节点对等和网络对等

集群管理员可以将**节点**和**网络**对等组合到 ANP 和 BANP 策略中。

例 2.10. 节点和网络对等示例

```yaml
apiVersion: policy.networking.k8s.io/v1alpha1
kind: AdminNetworkPolicy
metadata:
  name: egress-peer-1 1
spec:
  egress: 2
  - action: "Allow"
    name: "allow-egress"
    to:
    - nodes:
        matchExpressions:
        - key: worker-group
          operator: In
          values:
          - workloads # Egress traffic from nodes with label worker-group: workloads is allowed.
    - networks:
      - 104.154.164.170/32
    - pods:
        namespaceSelector:
          matchLabels:
            apps: external-apps
        podSelector:
          matchLabels:
            app: web # This rule in the policy allows the traffic directed to pods labeled apps: web in projects with apps: external-apps to leave the cluster.
  - action: "Deny"
    name: "deny-egress"
    to:
    - nodes:
        matchExpressions:
        - key: worker-group
          operator: In
          values:
          - infra # Egress traffic from nodes with label worker-group: infra is denied.
    - networks:
      - 104.154.164.160/32 # Egress traffic to this IP address from cluster is denied.
    - pods:
        namespaceSelector:
          matchLabels:
            apps: internal-apps
        podSelector: {}
  - action: "Pass"
    name: "pass-egress"
    to:
    - nodes:
        matchExpressions:
        - key: node-role.kubernetes.io/worker
          operator: Exists # All other egress traffic is passed to NetworkPolicy or BANP for evaluation.
  priority: 30 3
  subject: 4
    namespaces:
      matchLabels:
        apps: all-apps
```

1 指定策略的名称。

2. 对于**节点**和**网络**对等，您只能在 ANP 中使用北向流量控制作为 **egress**。

3. 指定 ANP 的优先级，确定应评估它们的顺序。低的优先级规则具有更高的优先权。ANP 接受 0-99 的值，0 为最高优先级，99 为最低优先级。

4. 指定集群中要应用策略的 pod 集合。在示例中，所有命名空间中带有 **apps: all-apps** 标签的任何 pod 都是策略的主题。

2.5. ADMINNETWORKPOLICY 故障排除

2.5.1. 检查 ANP 的创建

要检查您的 **AdminNetworkPolicy** (ANP) 和 **BaselineAdminNetworkPolicy** (BANP) 是否已正确创建，请检查以下命令的状态输出： **oc describe ap** 或 **oc describe banp**。

正常状态表示 **OVN DB plumbing was successful** 和 **SetupSucceeded**。

例 2.11. 具有良好状态的 ANP 示例

```
...
Conditions:
Last Transition Time:  2024-06-08T20:29:00Z
Message:               Setting up OVN DB plumbing was successful
Reason:                SetupSucceeded
Status:                True
Type:                  Ready-In-Zone-ovn-control-plane Last Transition Time:  2024-06-08T20:29:00Z
Message:               Setting up OVN DB plumbing was successful
Reason:                SetupSucceeded
Status:                True
Type:                  Ready-In-Zone-ovn-worker
Last Transition Time:  2024-06-08T20:29:00Z
Message:               Setting up OVN DB plumbing was successful
Reason:                SetupSucceeded
Status:                True
Type:                  Ready-In-Zone-ovn-worker2
...
```

如果 Plumbing 失败，则会从相应的区控制器报告错误。

例 2.12. 带有错误状态和错误消息的 ANP 示例

```
...
Status:
  Conditions:
    Last Transition Time:  2024-06-25T12:47:44Z
    Message:               error attempting to add ANP cluster-control with priority 600 because, OVNK
only supports priority ranges 0-99
    Reason:                SetupFailed
    Status:                False
    Type:                  Ready-In-Zone-example-worker-1.example.example-org.net
```

~~~
    Last Transition Time:  2024-06-25T12:47:45Z
    Message:               error attempting to add ANP cluster-control with priority 600 because, OVNK
only supports priority ranges 0-99
    Reason:                SetupFailed
    Status:                False
    Type:                  Ready-In-Zone-example-worker-0.example.example-org.net
    Last Transition Time:  2024-06-25T12:47:44Z
    Message:               error attempting to add ANP cluster-control with priority 600 because, OVNK
only supports priority ranges 0-99
    Reason:                SetupFailed
    Status:                False
    Type:                  Ready-In-Zone-example-ctlplane-1.example.example-org.net
    Last Transition Time:  2024-06-25T12:47:44Z
    Message:               error attempting to add ANP cluster-control with priority 600 because, OVNK
only supports priority ranges 0-99
    Reason:                SetupFailed
    Status:                False
    Type:                  Ready-In-Zone-example-ctlplane-2.example.example-org.net
    Last Transition Time:  2024-06-25T12:47:44Z
    Message:               error attempting to add ANP cluster-control with priority 600 because, OVNK
only supports priority ranges 0-99
    Reason:                SetupFailed
    Status:                False
    Type:                  Ready-In-Zone-example-ctlplane-0.example.example-org.net
    ```
~~~

有关 **nbctl** 命令，请参见以下部分来帮助排除不成功的策略。

2.5.1.1. 为 ANP 和 BANP 使用 nbctl 命令

要对不成功的设置进行故障排除，请首先查看 OVN 北向数据库 (nbdb) 对象，包括 **ACL**、**AdressSet** 和 **Port_Group**。要查看 nbdb，您需要在该节点上的 pod 内部查看该节点数据库中的对象。

先决条件

- 使用具有 **cluster-admin** 角色的用户访问集群。
- 已安装 OpenShift CLI（**oc**）。

注意

要在集群中运行 ovn **nbctl** 命令，您必须在相关节点的 "nbdb" 中打开远程 shell。

以下策略用于生成输出。

例 2.13. 用于生成输出的 AdminNetworkPolicy

```
apiVersion: policy.networking.k8s.io/v1alpha1
kind: AdminNetworkPolicy
metadata:
  name: cluster-control
spec:
  priority: 34
```

```
subject:
  namespaces:
    matchLabels:
      anp: cluster-control-anp # Only namespaces with this label have this ANP
ingress:
- name: "allow-from-ingress-router" # rule0
  action: "Allow"
  from:
  - namespaces:
      matchLabels:
        policy-group.network.openshift.io/ingress: ""
- name: "allow-from-monitoring" # rule1
  action: "Allow"
  from:
  - namespaces:
      matchLabels:
        kubernetes.io/metadata.name: openshift-monitoring
  ports:
  - portNumber:
      protocol: TCP
      port: 7564
  - namedPort: "scrape"
- name: "allow-from-open-tenants" # rule2
  action: "Allow"
  from:
  - namespaces: # open tenants
      matchLabels:
        tenant: open
- name: "pass-from-restricted-tenants" # rule3
  action: "Pass"
  from:
  - namespaces: # restricted tenants
      matchLabels:
        tenant: restricted
- name: "default-deny" # rule4
  action: "Deny"
  from:
  - namespaces: {} # Use the empty selector with caution because it also selects OpenShift namespaces as well.
egress:
- name: "allow-to-dns" # rule0
  action: "Allow"
  to:
  - pods:
      namespaceSelector:
        matchLabels:
          kubernetes.io/metadata.name: openshift-dns
      podSelector:
        matchLabels:
          app: dns
  ports:
  - portNumber:
      protocol: UDP
      port: 5353
- name: "allow-to-kapi-server" # rule1
  action: "Allow"
```

```
      to:
      - nodes:
          matchExpressions:
          - key: node-role.kubernetes.io/control-plane
            operator: Exists
      ports:
      - portNumber:
          protocol: TCP
          port: 6443
    - name: "allow-to-splunk" # rule2
      action: "Allow"
      to:
      - namespaces:
          matchLabels:
            tenant: splunk
      ports:
      - portNumber:
          protocol: TCP
          port: 8991
      - portNumber:
          protocol: TCP
          port: 8992
    - name: "allow-to-open-tenants-and-intranet-and-worker-nodes" # rule3
      action: "Allow"
      to:
      - nodes: # worker-nodes
          matchExpressions:
          - key: node-role.kubernetes.io/worker
            operator: Exists
      - networks: # intranet
        - 172.29.0.0/30
        - 10.0.54.0/19
        - 10.0.56.38/32
        - 10.0.69.0/24
      - namespaces: # open tenants
          matchLabels:
            tenant: open
    - name: "pass-to-restricted-tenants" # rule4
      action: "Pass"
      to:
      - namespaces: # restricted tenants
          matchLabels:
            tenant: restricted
    - name: "default-deny"
      action: "Deny"
      to:
      - networks:
        - 0.0.0.0/0
```

流程

1. 运行以下命令，使用节点信息列出 pod：

```
$ oc get pods -n openshift-ovn-kubernetes -owide
```

输出示例

```
NAME                                    READY   STATUS    RESTARTS   AGE   IP           NODE
NOMINATED NODE   READINESS GATES
ovnkube-control-plane-5c95487779-8k9fd   2/2     Running   0          34m   10.0.0.5     ci-ln-
0tv5gg2-72292-6sjw5-master-0          <none>           <none>
ovnkube-control-plane-5c95487779-v2xn8   2/2     Running   0          34m   10.0.0.3     ci-ln-
0tv5gg2-72292-6sjw5-master-1          <none>           <none>
ovnkube-node-524dt                       8/8     Running   0          33m   10.0.0.4     ci-ln-0tv5gg2-
72292-6sjw5-master-2          <none>           <none>
ovnkube-node-gbwr9                       8/8     Running   0          24m   10.0.128.4   ci-ln-0tv5gg2-
72292-6sjw5-worker-c-s9gqt   <none>           <none>
ovnkube-node-h4fpx                       8/8     Running   0          33m   10.0.0.5     ci-ln-0tv5gg2-
72292-6sjw5-master-0          <none>           <none>
ovnkube-node-j4hzw                       8/8     Running   0          24m   10.0.128.2   ci-ln-0tv5gg2-
72292-6sjw5-worker-a-hzbh5   <none>           <none>
ovnkube-node-wdhgv                       8/8     Running   0          33m   10.0.0.3     ci-ln-0tv5gg2-
72292-6sjw5-master-1          <none>           <none>
ovnkube-node-wfncn                       8/8     Running   0          24m   10.0.128.3   ci-ln-0tv5gg2-
72292-6sjw5-worker-b-5bb7f   <none>           <none>
```

2. 运行以下命令，进入 pod 以查看北向数据库：

```
$ oc rsh -c nbdb -n openshift-ovn-kubernetes ovnkube-node-524dt
```

3. 运行以下命令来查看 ACL nbdb ：

```
$ ovn-nbctl find ACL 'external_ids{>=}{"k8s.ovn.org/owner-
type"=AdminNetworkPolicy,"k8s.ovn.org/name"=cluster-control}'
```

其中, cluster-control
: 指定您要故障排除的 **AdminNetworkPolicy** 的名称。

AdminNetworkPolicy
: 指定类型: **AdminNetworkPolicy** 或 **BaselineAdminNetworkPolicy**。

例 2.14. ACL 的输出示例

```
_uuid               : 0d5e4722-b608-4bb1-b625-23c323cc9926
action              : allow-related
direction           : to-lport
external_ids        : {direction=Ingress, gress-index="2", "k8s.ovn.org/id"="default-network-
controller:AdminNetworkPolicy:cluster-control:Ingress:2:None",
"k8s.ovn.org/name"=cluster-control, "k8s.ovn.org/owner-controller"=default-network-
controller, "k8s.ovn.org/owner-type"=AdminNetworkPolicy, port-policy-protocol=None}
label               : 0
log                 : false
match               : "outport == @a14645450421485494999 && ((ip4.src ==
$a13730899355151937870))"
meter               : acl-logging
name                : "ANP:cluster-control:Ingress:2"
options             : {}
priority            : 26598
severity            : []
```

```
tier              : 1

_uuid             : b7be6472-df67-439c-8c9c-f55929f0a6e0
action            : drop
direction         : from-lport
external_ids      : {direction=Egress, gress-index="5", "k8s.ovn.org/id"="default-network-
controller:AdminNetworkPolicy:cluster-control:Egress:5:None",
"k8s.ovn.org/name"=cluster-control, "k8s.ovn.org/owner-controller"=default-network-
controller, "k8s.ovn.org/owner-type"=AdminNetworkPolicy, port-policy-protocol=None}
label             : 0
log               : false
match             : "inport == @a14645450421485494999 && ((ip4.dst ==
$a11452480169090787059))"
meter             : acl-logging
name              : "ANP:cluster-control:Egress:5"
options           : {apply-after-lb="true"}
priority          : 26595
severity          : []
tier              : 1

_uuid             : 5a6e5bb4-36eb-4209-b8bc-c611983d4624
action            : pass
direction         : to-lport
external_ids      : {direction=Ingress, gress-index="3", "k8s.ovn.org/id"="default-network-
controller:AdminNetworkPolicy:cluster-control:Ingress:3:None",
"k8s.ovn.org/name"=cluster-control, "k8s.ovn.org/owner-controller"=default-network-
controller, "k8s.ovn.org/owner-type"=AdminNetworkPolicy, port-policy-protocol=None}
label             : 0
log               : false
match             : "outport == @a14645450421485494999 && ((ip4.src ==
$a764182844364804195))"
meter             : acl-logging
name              : "ANP:cluster-control:Ingress:3"
options           : {}
priority          : 26597
severity          : []
tier              : 1

_uuid             : 04f20275-c410-405c-a923-0e677f767889
action            : pass
direction         : from-lport
external_ids      : {direction=Egress, gress-index="4", "k8s.ovn.org/id"="default-network-
controller:AdminNetworkPolicy:cluster-control:Egress:4:None",
"k8s.ovn.org/name"=cluster-control, "k8s.ovn.org/owner-controller"=default-network-
controller, "k8s.ovn.org/owner-type"=AdminNetworkPolicy, port-policy-protocol=None}
label             : 0
log               : false
match             : "inport == @a14645450421485494999 && ((ip4.dst ==
$a5972452606168369118))"
meter             : acl-logging
name              : "ANP:cluster-control:Egress:4"
options           : {apply-after-lb="true"}
priority          : 26596
severity          : []
tier              : 1
```

```
_uuid               : 4b5d836a-e0a3-4088-825e-f9f0ca58e538
action              : drop
direction           : to-lport
external_ids        : {direction=Ingress, gress-index="4", "k8s.ovn.org/id"="default-network-controller:AdminNetworkPolicy:cluster-control:Ingress:4:None", "k8s.ovn.org/name"=cluster-control, "k8s.ovn.org/owner-controller"=default-network-controller, "k8s.ovn.org/owner-type"=AdminNetworkPolicy, port-policy-protocol=None}
label               : 0
log                 : false
match               : "outport == @a14645450421485494999 && ((ip4.src == $a13814616246365836720))"
meter               : acl-logging
name                : "ANP:cluster-control:Ingress:4"
options             : {}
priority            : 26596
severity            : []
tier                : 1

_uuid               : 5d09957d-d2cc-4f5a-9ddd-b97d9d772023
action              : allow-related
direction           : from-lport
external_ids        : {direction=Egress, gress-index="2", "k8s.ovn.org/id"="default-network-controller:AdminNetworkPolicy:cluster-control:Egress:2:tcp", "k8s.ovn.org/name"=cluster-control, "k8s.ovn.org/owner-controller"=default-network-controller, "k8s.ovn.org/owner-type"=AdminNetworkPolicy, port-policy-protocol=tcp}
label               : 0
log                 : false
match               : "inport == @a14645450421485494999 && ((ip4.dst == $a18396736153283155648)) && tcp && tcp.dst=={8991,8992}"
meter               : acl-logging
name                : "ANP:cluster-control:Egress:2"
options             : {apply-after-lb="true"}
priority            : 26598
severity            : []
tier                : 1

_uuid               : 1a68a5ed-e7f9-47d0-b55c-89184d97e81a
action              : allow-related
direction           : from-lport
external_ids        : {direction=Egress, gress-index="1", "k8s.ovn.org/id"="default-network-controller:AdminNetworkPolicy:cluster-control:Egress:1:tcp", "k8s.ovn.org/name"=cluster-control, "k8s.ovn.org/owner-controller"=default-network-controller, "k8s.ovn.org/owner-type"=AdminNetworkPolicy, port-policy-protocol=tcp}
label               : 0
log                 : false
match               : "inport == @a14645450421485494999 && ((ip4.dst == $a10706246167277696183)) && tcp && tcp.dst==6443"
meter               : acl-logging
name                : "ANP:cluster-control:Egress:1"
options             : {apply-after-lb="true"}
priority            : 26599
severity            : []
tier                : 1

_uuid               : aa1a224d-7960-4952-bdfb-35246bafbac8
action              : allow-related
```

```
direction           : to-lport
external_ids        : {direction=Ingress, gress-index="1", "k8s.ovn.org/id"="default-network-controller:AdminNetworkPolicy:cluster-control:Ingress:1:tcp", "k8s.ovn.org/name"=cluster-control, "k8s.ovn.org/owner-controller"=default-network-controller, "k8s.ovn.org/owner-type"=AdminNetworkPolicy, port-policy-protocol=tcp}
label               : 0
log                 : false
match               : "outport == @a14645450421485494999 && ((ip4.src == $a6786643370959569281)) && tcp && tcp.dst==7564"
meter               : acl-logging
name                : "ANP:cluster-control:Ingress:1"
options             : {}
priority            : 26599
severity            : []
tier                : 1

_uuid               : 1a27d30e-3f96-4915-8ddd-ade7f22c117b
action              : allow-related
direction           : from-lport
external_ids        : {direction=Egress, gress-index="3", "k8s.ovn.org/id"="default-network-controller:AdminNetworkPolicy:cluster-control:Egress:3:None", "k8s.ovn.org/name"=cluster-control, "k8s.ovn.org/owner-controller"=default-network-controller, "k8s.ovn.org/owner-type"=AdminNetworkPolicy, port-policy-protocol=None}
label               : 0
log                 : false
match               : "inport == @a14645450421485494999 && ((ip4.dst == $a10622494091691694581))"
meter               : acl-logging
name                : "ANP:cluster-control:Egress:3"
options             : {apply-after-lb="true"}
priority            : 26597
severity            : []
tier                : 1

_uuid               : b23a087f-08f8-4225-8c27-4a9a9ee0c407
action              : allow-related
direction           : from-lport
external_ids        : {direction=Egress, gress-index="0", "k8s.ovn.org/id"="default-network-controller:AdminNetworkPolicy:cluster-control:Egress:0:udp", "k8s.ovn.org/name"=cluster-control, "k8s.ovn.org/owner-controller"=default-network-controller, "k8s.ovn.org/owner-type"=AdminNetworkPolicy, port-policy-protocol=udp}
label               : 0
log                 : false
match               : "inport == @a14645450421485494999 && ((ip4.dst == $a13517855690389298082)) && udp && udp.dst==5353"
meter               : acl-logging
name                : "ANP:cluster-control:Egress:0"
options             : {apply-after-lb="true"}
priority            : 26600
severity            : []
tier                : 1

_uuid               : d14ed5cf-2e06-496e-8cae-6b76d5dd5ccd
action              : allow-related
direction           : to-lport
external_ids        : {direction=Ingress, gress-index="0", "k8s.ovn.org/id"="default-network-
```

```
controller:AdminNetworkPolicy:cluster-control:Ingress:0:None",
"k8s.ovn.org/name"=cluster-control, "k8s.ovn.org/owner-controller"=default-network-
controller, "k8s.ovn.org/owner-type"=AdminNetworkPolicy, port-policy-protocol=None}
label               : 0
log                 : false
match               : "outport == @a14645450421485494999 && ((ip4.src ==
$a14545668191619617708))"
meter               : acl-logging
name                : "ANP:cluster-control:Ingress:0"
options             : {}
priority            : 26600
severity            : []
tier                : 1
```

注意

ingress 和 egress 的输出显示策略在 ACL 中的逻辑。例如，每次数据包与提供的 **match** 匹配时会执行的 **action**。

a. 运行以下命令，为规则检查特定 ACL：

```
$ ovn-nbctl find ACL 'external_ids{>=}{"k8s.ovn.org/owner-
type"=AdminNetworkPolicy,direction=Ingress,"k8s.ovn.org/name"=cluster-control,gress-
index="1"}'
```

其中, cluster-control

指定 ANP 的**名称**。

入口

指定流量的 **direction** 为类型 **Ingress** 或 **Egress**。

1

指定要查看的规则。

对于示名为 **cluster-control** 的 ANP 示例，其 **priority** 是 **34**，以下是 **Ingress rule** 1 的示例输出：

例 2.15. 输出示例

```
_uuid               : aa1a224d-7960-4952-bdfb-35246bafbac8
action              : allow-related
direction           : to-lport
external_ids        : {direction=Ingress, gress-index="1", "k8s.ovn.org/id"="default-
network-controller:AdminNetworkPolicy:cluster-control:Ingress:1:tcp",
"k8s.ovn.org/name"=cluster-control, "k8s.ovn.org/owner-controller"=default-network-
controller, "k8s.ovn.org/owner-type"=AdminNetworkPolicy, port-policy-protocol=tcp}
label               : 0
log                 : false
match               : "outport == @a14645450421485494999 && ((ip4.src ==
$a6786643370959569281)) && tcp && tcp.dst==7564"
meter               : acl-logging
name                : "ANP:cluster-control:Ingress:1"
options             : {}
```

```
priority          : 26599
severity          : []
tier              : 1
```

4. 运行以下命令查看 nbdb 中的地址集：

```
$ ovn-nbctl find Address_Set 'external_ids{>=}{"k8s.ovn.org/owner-type"=AdminNetworkPolicy,"k8s.ovn.org/name"=cluster-control}'
```

例 2.16. Address_Set 的输出示例

```
_uuid               : 56e89601-5552-4238-9fc3-8833f5494869
addresses           : ["192.168.194.135", "192.168.194.152", "192.168.194.193", "192.168.194.254"]
external_ids        : {direction=Egress, gress-index="1", ip-family=v4, "k8s.ovn.org/id"="default-network-controller:AdminNetworkPolicy:cluster-control:Egress:1:v4", "k8s.ovn.org/name"=cluster-control, "k8s.ovn.org/owner-controller"=default-network-controller, "k8s.ovn.org/owner-type"=AdminNetworkPolicy}
name                : a10706246167277696183

_uuid               : 7df9330d-380b-4bdb-8acd-4eddeda2419c
addresses           : ["10.132.0.10", "10.132.0.11", "10.132.0.12", "10.132.0.13", "10.132.0.14", "10.132.0.15", "10.132.0.16", "10.132.0.17", "10.132.0.5", "10.132.0.7", "10.132.0.71", "10.132.0.75", "10.132.0.8", "10.132.0.81", "10.132.0.9", "10.132.2.10", "10.132.2.11", "10.132.2.12", "10.132.2.14", "10.132.2.15", "10.132.2.3", "10.132.2.4", "10.132.2.5", "10.132.2.6", "10.132.2.7", "10.132.2.8", "10.132.2.9", "10.132.3.64", "10.132.3.65", "10.132.3.72", "10.132.3.73", "10.132.3.76", "10.133.0.10", "10.133.0.11", "10.133.0.12", "10.133.0.13", "10.133.0.14", "10.133.0.15", "10.133.0.16", "10.133.0.17", "10.133.0.18", "10.133.0.19", "10.133.0.20", "10.133.0.21", "10.133.0.22", "10.133.0.23", "10.133.0.24", "10.133.0.25", "10.133.0.26", "10.133.0.27", "10.133.0.28", "10.133.0.29", "10.133.0.30", "10.133.0.31", "10.133.0.32", "10.133.0.33", "10.133.0.34", "10.133.0.35", "10.133.0.36", "10.133.0.37", "10.133.0.38", "10.133.0.39", "10.133.0.40", "10.133.0.41", "10.133.0.42", "10.133.0.44", "10.133.0.45", "10.133.0.46", "10.133.0.47", "10.133.0.48", "10.133.0.5", "10.133.0.6", "10.133.0.7", "10.133.0.8", "10.133.0.9", "10.134.0.10", "10.134.0.11", "10.134.0.12", "10.134.0.13", "10.134.0.14", "10.134.0.15", "10.134.0.16", "10.134.0.17", "10.134.0.18", "10.134.0.19", "10.134.0.20", "10.134.0.21", "10.134.0.22", "10.134.0.23", "10.134.0.24", "10.134.0.25", "10.134.0.26", "10.134.0.27", "10.134.0.28", "10.134.0.30", "10.134.0.31", "10.134.0.32", "10.134.0.33", "10.134.0.34", "10.134.0.35", "10.134.0.36", "10.134.0.37", "10.134.0.38", "10.134.0.4", "10.134.0.42", "10.134.0.9", "10.135.0.10", "10.135.0.11", "10.135.0.12", "10.135.0.13", "10.135.0.14", "10.135.0.15", "10.135.0.16", "10.135.0.17", "10.135.0.18", "10.135.0.19", "10.135.0.23", "10.135.0.24", "10.135.0.26", "10.135.0.27", "10.135.0.29", "10.135.0.3", "10.135.0.4", "10.135.0.40", "10.135.0.41", "10.135.0.42", "10.135.0.43", "10.135.0.44", "10.135.0.5", "10.135.0.6", "10.135.0.7", "10.135.0.8", "10.135.0.9"]
external_ids        : {direction=Ingress, gress-index="4", ip-family=v4, "k8s.ovn.org/id"="default-network-controller:AdminNetworkPolicy:cluster-control:Ingress:4:v4", "k8s.ovn.org/name"=cluster-control, "k8s.ovn.org/owner-controller"=default-network-controller, "k8s.ovn.org/owner-type"=AdminNetworkPolicy}
name                : a13814616246365836720

_uuid               : 84d76f13-ad95-4c00-8329-a0b1d023c289
addresses           : ["10.132.3.76", "10.135.0.44"]
external_ids        : {direction=Egress, gress-index="4", ip-family=v4, "k8s.ovn.org/id"="default-network-controller:AdminNetworkPolicy:cluster-
```

```
control:Egress:4:v4", "k8s.ovn.org/name"=cluster-control, "k8s.ovn.org/owner-
controller"=default-network-controller, "k8s.ovn.org/owner-type"=AdminNetworkPolicy}
name                : a5972452606168369118

_uuid               : 0c53e917-f7ee-4256-8f3a-9522c0481e52
addresses           : ["10.132.0.10", "10.132.0.11", "10.132.0.12", "10.132.0.13",
"10.132.0.14", "10.132.0.15", "10.132.0.16", "10.132.0.17", "10.132.0.5", "10.132.0.7",
"10.132.0.71", "10.132.0.75", "10.132.0.8", "10.132.0.81", "10.132.0.9", "10.132.2.10",
"10.132.2.11", "10.132.2.12", "10.132.2.14", "10.132.2.15", "10.132.2.3", "10.132.2.4",
"10.132.2.5", "10.132.2.6", "10.132.2.7", "10.132.2.8", "10.132.2.9", "10.132.3.64",
"10.132.3.65", "10.132.3.72", "10.132.3.73", "10.132.3.76", "10.133.0.10", "10.133.0.11",
"10.133.0.12", "10.133.0.13", "10.133.0.14", "10.133.0.15", "10.133.0.16", "10.133.0.17",
"10.133.0.18", "10.133.0.19", "10.133.0.20", "10.133.0.21", "10.133.0.22", "10.133.0.23",
"10.133.0.24", "10.133.0.25", "10.133.0.26", "10.133.0.27", "10.133.0.28", "10.133.0.29",
"10.133.0.30", "10.133.0.31", "10.133.0.32", "10.133.0.33", "10.133.0.34", "10.133.0.35",
"10.133.0.36", "10.133.0.37", "10.133.0.38", "10.133.0.39", "10.133.0.40", "10.133.0.41",
"10.133.0.42", "10.133.0.44", "10.133.0.45", "10.133.0.46", "10.133.0.47", "10.133.0.48",
"10.133.0.5", "10.133.0.6", "10.133.0.7", "10.133.0.8", "10.133.0.9", "10.134.0.10",
"10.134.0.11", "10.134.0.12", "10.134.0.13", "10.134.0.14", "10.134.0.15", "10.134.0.16",
"10.134.0.17", "10.134.0.18", "10.134.0.19", "10.134.0.20", "10.134.0.21", "10.134.0.22",
"10.134.0.23", "10.134.0.24", "10.134.0.25", "10.134.0.26", "10.134.0.27", "10.134.0.28",
"10.134.0.30", "10.134.0.31", "10.134.0.32", "10.134.0.33", "10.134.0.34", "10.134.0.35",
"10.134.0.36", "10.134.0.37", "10.134.0.38", "10.134.0.4", "10.134.0.42", "10.134.0.9",
"10.135.0.10", "10.135.0.11", "10.135.0.12", "10.135.0.13", "10.135.0.14", "10.135.0.15",
"10.135.0.16", "10.135.0.17", "10.135.0.18", "10.135.0.19", "10.135.0.23", "10.135.0.24",
"10.135.0.26", "10.135.0.27", "10.135.0.29", "10.135.0.3", "10.135.0.4", "10.135.0.40",
"10.135.0.41", "10.135.0.42", "10.135.0.43", "10.135.0.44", "10.135.0.5", "10.135.0.6",
"10.135.0.7", "10.135.0.8", "10.135.0.9"]
external_ids        : {direction=Egress, gress-index="2", ip-family=v4,
"k8s.ovn.org/id"="default-network-controller:AdminNetworkPolicy:cluster-
control:Egress:2:v4", "k8s.ovn.org/name"=cluster-control, "k8s.ovn.org/owner-
controller"=default-network-controller, "k8s.ovn.org/owner-type"=AdminNetworkPolicy}
name                : a18396736153283155648

_uuid               : 5228bf1b-dfd8-40ec-bfa8-95c5bf9aded9
addresses           : []
external_ids        : {direction=Ingress, gress-index="0", ip-family=v4,
"k8s.ovn.org/id"="default-network-controller:AdminNetworkPolicy:cluster-
control:Ingress:0:v4", "k8s.ovn.org/name"=cluster-control, "k8s.ovn.org/owner-
controller"=default-network-controller, "k8s.ovn.org/owner-type"=AdminNetworkPolicy}
name                : a14545668191619617708

_uuid               : 46530d69-70da-4558-8c63-884ec9dc4f25
addresses           : ["10.132.2.10", "10.132.2.5", "10.132.2.6", "10.132.2.7", "10.132.2.8",
"10.132.2.9", "10.133.0.47", "10.134.0.33", "10.135.0.10", "10.135.0.11", "10.135.0.12",
"10.135.0.19", "10.135.0.24", "10.135.0.7", "10.135.0.8", "10.135.0.9"]
external_ids        : {direction=Ingress, gress-index="1", ip-family=v4,
"k8s.ovn.org/id"="default-network-controller:AdminNetworkPolicy:cluster-
control:Ingress:1:v4", "k8s.ovn.org/name"=cluster-control, "k8s.ovn.org/owner-
controller"=default-network-controller, "k8s.ovn.org/owner-type"=AdminNetworkPolicy}
name                : a6786643370959569281

_uuid               : 65fdcdea-0b9f-4318-9884-1b51d231ad1d
addresses           : ["10.132.3.72", "10.135.0.42"]
external_ids        : {direction=Ingress, gress-index="2", ip-family=v4,
"k8s.ovn.org/id"="default-network-controller:AdminNetworkPolicy:cluster-
```

```
control:Ingress:2:v4", "k8s.ovn.org/name"=cluster-control, "k8s.ovn.org/owner-
controller"=default-network-controller, "k8s.ovn.org/owner-type"=AdminNetworkPolicy}
name                : a13730899355151937870

_uuid               : 73eabdb0-36bf-4ca3-b66d-156ac710df4c
addresses           : ["10.0.32.0/19", "10.0.56.38/32", "10.0.69.0/24", "10.132.3.72",
"10.135.0.42", "172.29.0.0/30", "192.168.194.103", "192.168.194.2"]
external_ids        : {direction=Egress, gress-index="3", ip-family=v4,
"k8s.ovn.org/id"="default-network-controller:AdminNetworkPolicy:cluster-
control:Egress:3:v4", "k8s.ovn.org/name"=cluster-control, "k8s.ovn.org/owner-
controller"=default-network-controller, "k8s.ovn.org/owner-type"=AdminNetworkPolicy}
name                : a10622494091691694581

_uuid               : 50cdbef2-71b5-474b-914c-6fcd1d7712d3
addresses           : ["10.132.0.10", "10.132.0.11", "10.132.0.12", "10.132.0.13",
"10.132.0.14", "10.132.0.15", "10.132.0.16", "10.132.0.17", "10.132.0.5", "10.132.0.7",
"10.132.0.71", "10.132.0.75", "10.132.0.8", "10.132.0.81", "10.132.0.9", "10.132.2.10",
"10.132.2.11", "10.132.2.12", "10.132.2.14", "10.132.2.15", "10.132.2.3", "10.132.2.4",
"10.132.2.5", "10.132.2.6", "10.132.2.7", "10.132.2.8", "10.132.2.9", "10.132.3.64",
"10.132.3.65", "10.132.3.72", "10.132.3.73", "10.132.3.76", "10.133.0.10", "10.133.0.11",
"10.133.0.12", "10.133.0.13", "10.133.0.14", "10.133.0.15", "10.133.0.16", "10.133.0.17",
"10.133.0.18", "10.133.0.19", "10.133.0.20", "10.133.0.21", "10.133.0.22", "10.133.0.23",
"10.133.0.24", "10.133.0.25", "10.133.0.26", "10.133.0.27", "10.133.0.28", "10.133.0.29",
"10.133.0.30", "10.133.0.31", "10.133.0.32", "10.133.0.33", "10.133.0.34", "10.133.0.35",
"10.133.0.36", "10.133.0.37", "10.133.0.38", "10.133.0.39", "10.133.0.40", "10.133.0.41",
"10.133.0.42", "10.133.0.44", "10.133.0.45", "10.133.0.46", "10.133.0.47", "10.133.0.48",
"10.133.0.5", "10.133.0.6", "10.133.0.7", "10.133.0.8", "10.133.0.9", "10.134.0.10",
"10.134.0.11", "10.134.0.12", "10.134.0.13", "10.134.0.14", "10.134.0.15", "10.134.0.16",
"10.134.0.17", "10.134.0.18", "10.134.0.19", "10.134.0.20", "10.134.0.21", "10.134.0.22",
"10.134.0.23", "10.134.0.24", "10.134.0.25", "10.134.0.26", "10.134.0.27", "10.134.0.28",
"10.134.0.30", "10.134.0.31", "10.134.0.32", "10.134.0.33", "10.134.0.34", "10.134.0.35",
"10.134.0.36", "10.134.0.37", "10.134.0.38", "10.134.0.4", "10.134.0.42", "10.134.0.9",
"10.135.0.10", "10.135.0.11", "10.135.0.12", "10.135.0.13", "10.135.0.14", "10.135.0.15",
"10.135.0.16", "10.135.0.17", "10.135.0.18", "10.135.0.19", "10.135.0.23", "10.135.0.24",
"10.135.0.26", "10.135.0.27", "10.135.0.29", "10.135.0.3", "10.135.0.4", "10.135.0.40",
"10.135.0.41", "10.135.0.42", "10.135.0.43", "10.135.0.44", "10.135.0.5", "10.135.0.6",
"10.135.0.7", "10.135.0.8", "10.135.0.9"]
external_ids        : {direction=Egress, gress-index="0", ip-family=v4,
"k8s.ovn.org/id"="default-network-controller:AdminNetworkPolicy:cluster-
control:Egress:0:v4", "k8s.ovn.org/name"=cluster-control, "k8s.ovn.org/owner-
controller"=default-network-controller, "k8s.ovn.org/owner-type"=AdminNetworkPolicy}
name                : a13517855690389298082

_uuid               : 32a42f32-2d11-43dd-979d-a56d7ee6aa57
addresses           : ["10.132.3.76", "10.135.0.44"]
external_ids        : {direction=Ingress, gress-index="3", ip-family=v4,
"k8s.ovn.org/id"="default-network-controller:AdminNetworkPolicy:cluster-
control:Ingress:3:v4", "k8s.ovn.org/name"=cluster-control, "k8s.ovn.org/owner-
controller"=default-network-controller, "k8s.ovn.org/owner-type"=AdminNetworkPolicy}
name                : a764182844364804195

_uuid               : 8fd3b977-6e1c-47aa-82b7-e3e3136c4a72
addresses           : ["0.0.0.0/0"]
external_ids        : {direction=Egress, gress-index="5", ip-family=v4,
"k8s.ovn.org/id"="default-network-controller:AdminNetworkPolicy:cluster-
```

```
control:Egress:5:v4", "k8s.ovn.org/name"=cluster-control, "k8s.ovn.org/owner-controller"=default-network-controller, "k8s.ovn.org/owner-type"=AdminNetworkPolicy}
name                : a11452480169090787059
```

a. 运行以下命令，检查规则的具体地址集：

```
$ ovn-nbctl find Address_Set 'external_ids{>=}{"k8s.ovn.org/owner-type"=AdminNetworkPolicy,direction=Egress,"k8s.ovn.org/name"=cluster-control,gress-index="5"}'
```

例 2.17. Address_Set 的输出示例

```
_uuid               : 8fd3b977-6e1c-47aa-82b7-e3e3136c4a72
addresses           : ["0.0.0.0/0"]
external_ids        : {direction=Egress, gress-index="5", ip-family=v4, "k8s.ovn.org/id"="default-network-controller:AdminNetworkPolicy:cluster-control:Egress:5:v4", "k8s.ovn.org/name"=cluster-control, "k8s.ovn.org/owner-controller"=default-network-controller, "k8s.ovn.org/owner-type"=AdminNetworkPolicy}
name                : a11452480169090787059
```

5. 运行以下命令查看 nbdb 中的端口组：

```
$ ovn-nbctl find Port_Group 'external_ids{>=}{"k8s.ovn.org/owner-type"=AdminNetworkPolicy,"k8s.ovn.org/name"=cluster-control}'
```

例 2.18. Port_Group 的输出示例

```
_uuid               : f50acf71-7488-4b9a-b7b8-c8a024e99d21
acls                : [04f20275-c410-405c-a923-0e677f767889, 0d5e4722-b608-4bb1-b625-23c323cc9926, 1a27d30e-3f96-4915-8ddd-ade7f22c117b, 1a68a5ed-e7f9-47d0-b55c-89184d97e81a, 4b5d836a-e0a3-4088-825e-f9f0ca58e538, 5a6e5bb4-36eb-4209-b8bc-c611983d4624, 5d09957d-d2cc-4f5a-9ddd-b97d9d772023, aa1a224d-7960-4952-bdfb-35246bafbac8, b23a087f-08f8-4225-8c27-4a9a9ee0c407, b7be6472-df67-439c-8c9c-f55929f0a6e0, d14ed5cf-2e06-496e-8cae-6b76d5dd5ccd]
external_ids        : {"k8s.ovn.org/id"="default-network-controller:AdminNetworkPolicy:cluster-control", "k8s.ovn.org/name"=cluster-control, "k8s.ovn.org/owner-controller"=default-network-controller, "k8s.ovn.org/owner-type"=AdminNetworkPolicy}
name                : a14645450421485494999
ports               : [5e75f289-8273-4f8a-8798-8c10f7318833, de7e1b71-6184-445d-93e7-b20acadf41ea]
```

2.5.2. 其他资源

- 使用 ovnkube-trace 追踪 Openflow
- OVN-Kubernetes 故障排除

2.6. ADMINNETWORKPOLICY 的最佳实践

本节为 **AdminNetworkPolicy** 和 **BaselineAdminNetworkPolicy** 资源提供最佳实践。

2.6.1. 设计 AdminNetworkPolicy

在构建 **AdminNetworkPolicy** (ANP) 资源时，您可能会在创建策略时考虑以下内容：

- 您可以创建具有相同优先级的 ANP。如果您使用同一优先级创建两个 ANP，请确保它们不会将重叠规则应用到同一流量。每个值仅有一个规则会被应用，当相同的优先级值有多个规则时，不会保证会应用哪个规则。因为当创建了重叠的 ANP 时，无法保证哪些策略会被优先使用，所以请使用不同的优先级来设置 ANPs。
- 管理员必须创建应用于用户命名空间的 ANP，而不是系统命名空间。

> **重要**
>
> 不支持将 ANP 和 **BaselineAdminNetworkPolicy** (BANP) 应用到系统命名空间（**default**, **kube-system**, 任何以 **openshift-** 开头的命名空间），这样做会使集群无响应并处于无法正常工作的状态。

- 因为支持的优先级范围是 **0-100**，所以您可以将您的 ANP 设置为使用一个中间范围，如 **30-70**。这样就可以为这个范围外保留一些更高和更低的优先级占位符。即使在这个中间范围内，您可能也会希望保留一些空缺，以便随着基础架构的变化，您可以根据需要以正确的优先级插入新的 ANP。如果您打包了 ANP，则可能需要重新创建它们，以适应将来的任何更改。
- 当使用 **0.0.0.0/0** 或 **::/0** 创建一个强的 **Deny** 策略时，请确保基本流量具有较高优先级 **Allow** 或 **Pass** 规则。
- 当您需要确保一个连接无论在什么情况下都被允许时，使用 **Allow** 作为您的 **action** 字段。ANP 中的 **Allow** 规则代表连接始终被允许，**NetworkPolicy** 将被忽略。
- 使用 **Pass** 作为您的 **action** 字段，将允许或拒绝连接策略委托给 **NetworkPolicy** 层。
- 确保多个规则中的选择器不会重叠，因此同一 IP 不会出现在多个策略中，这可能导致性能和扩展限制。
- 避免将 **namedPorts** 与 **PortNumber** 和 **PortRange** 结合使用，因为这样会创建 6 个 ACL，并导致集群中的效率降低。

2.6.1.1. 使用 BaselineAdminNetworkPolicy 的注意事项

- 您只能在集群中定义单个 **BaselineAdminNetworkPolicy** (BANP) 资源。以下支持用于 BANP，管理员可能在设计其 BANP 中考虑：
  - 您可以在用户命名空间中为 cluster-local ingress 设置默认的拒绝策略。此 BANP 将强制开发人员添加 **NetworkPolicy** 对象来允许他们允许的入口流量，如果他们没有为 ingress 添加网络策略，则会被拒绝。
  - 您可以在用户命名空间中为 cluster-local egress 设置默认拒绝策略。此 BANP 将强制开发人员添加 **NetworkPolicy** 对象来允许他们允许的入口流量，如果他们没有为 ingress 添加网络策略，则会被拒绝。
  - 您可以为出口设置到集群内 DNS 服务的默认允许策略。此类 BANP 可确保命名空间的用户不必将允许出口 **NetworkPolicy** 设置为集群内 DNS 服务。
  - 您可以设置一个出口策略，允许内部出口流量到所有容器集，但拒绝访问所有外部端点（例

如 **0.0.0.0/0** 和 **::/0**）。此 BANP 允许用户工作负载向其他集群端点发送流量，但默认不发送到外部端点。然后，开发人员可以使用 **NetworkPolicy** 来允许其应用程序将流量发送到一组明确的外部服务。

- 确保对 BANP 设置了范围，以便它只拒绝到用户命名空间的流量，而不是系统命名空间。这是因为系统命名空间没有 **NetworkPolicy** 对象来覆盖 BANP。

2.6.1.2. AdminNetworkPolicy 和 NetworkPolicy 之间的不同

- 与 **NetworkPolicy** 对象不同，您必须使用显式标签在 ANP 和 BANP 中引用工作负载，而不是使用空 (**{}**) 捕获所有选择器以避免意外选择流量。

重要

应用到基础架构命名空间的空命名空间选择器可能会导致集群无响应且处于无法正常工作状态。

- 在 ANP 的 API 语义中，您必须在创建策略时显式定义允许或拒绝规则，这与具有隐式 deny 的 **NetworkPolicy** 对象不同。

- 与 **NetworkPolicy** 对象不同，**AdminNetworkPolicy** 对象入口规则仅限于集群内 Pod 和命名空间，因此您无法不需要，从主机网络为 ingress 设置规则。

第 3 章 网络策略

3.1. 关于网络策略

作为开发者，您可以定义网络策略来限制集群中 pod 的流量。

3.1.1. 关于网络策略

默认情况下，项目中的所有 pod 都可被其他 pod 和网络端点访问。要在一个项目中隔离一个或多个 Pod，您可以在该项目中创建 **NetworkPolicy** 对象来指示允许的入站连接。项目管理员可以在自己的项目中创建和删除 **NetworkPolicy** 对象。

如果一个 pod 由一个或多个 **NetworkPolicy** 对象中的选择器匹配，那么该 pod 将只接受至少被其中一个 **NetworkPolicy** 对象所允许的连接。未被任何 **NetworkPolicy** 对象选择的 pod 可以完全访问。

网络策略仅适用于传输控制协议(TCP)、用户数据报协议(UDP)、互联网控制消息协议(ICMP)和流控制传输协议(SCTP)协议。其他协议不会受到影响。

警告

- 网络策略不适用于主机网络命名空间。启用主机网络的 Pod 不受网络策略规则的影响。但是，连接到 host-networked pod 的 pod 会受到网络策略规则的影响。
- 使用没有将 **podSelector** 字段设置为 **{}** 的 **namespaceSelector** 字段将不会包括 **hostNetwork** pod。您必须使用 **namespaceSelector** 字段，**podSelector** 设置为 **{}**，以便在创建网络策略时目标 **hostNetwork** pod。
- 网络策略无法阻止来自 localhost 或来自其驻留的节点的流量。

以下示例 **NetworkPolicy** 对象演示了支持不同的情景：

- 拒绝所有流量：
  要使项目默认为拒绝流量，请添加一个匹配所有 pod 但不接受任何流量的 **NetworkPolicy** 对象：

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

- 只允许 OpenShift Container Platform Ingress Controller 的连接：
  要使项目只允许 OpenShift Container Platform Ingress Controller 的连接，请添加以下 **NetworkPolicy** 对象。

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-from-openshift-ingress
spec:
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          policy-group.network.openshift.io/ingress: ""
  podSelector: {}
  policyTypes:
  - Ingress
```

- 只接受项目中 pod 的连接：

重要

要允许同一命名空间中的 **hostNetwork** pod 的入站连接，您需要将 **allow-from-hostnetwork** 策略与 **allow-same-namespace** 策略一起应用。

要使 pod 接受同一项目中其他 pod 的连接，但拒绝其他项目中所有 pod 的连接，请添加以下 **NetworkPolicy** 对象：

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

- 仅允许基于 pod 标签的 HTTP 和 HTTPS 流量：
要对带有特定标签（以下示例中的 **role=frontend**）的 pod 仅启用 HTTP 和 HTTPS 访问，请添加类似如下的 **NetworkPolicy** 对象：

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: allow-http-and-https
spec:
  podSelector:
    matchLabels:
      role: frontend
  ingress:
  - ports:
    - protocol: TCP
      port: 80
    - protocol: TCP
      port: 443
```

- 使用命名空间和 pod 选择器接受连接：
  要通过组合使用命名空间和 pod 选择器来匹配网络流量,您可以使用类似如下的 **NetworkPolicy** 对象：

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: allow-pod-and-namespace-both
spec:
  podSelector:
    matchLabels:
      name: test-pods
  ingress:
    - from:
      - namespaceSelector:
          matchLabels:
            project: project_name
        podSelector:
          matchLabels:
            name: test-pods
```

NetworkPolicy 对象是可添加的；也就是说，您可以组合多个 **NetworkPolicy** 对象来满足复杂的网络要求。

例如，对于以上示例中定义的 **NetworkPolicy** 对象，您可以在同一个项目中定义 **allow-same-namespace** 和 **allow-http-and-https** 策略。因此，允许带有标签 **role=frontend** 的 pod 接受每一策略所允许的任何连接。即，任何端口上来自同一命名空间中的 pod 的连接，以及端口 **80** 和 **443** 上的来自任意命名空间中 pod 的连接。

3.1.1.1. 使用 allow-from-router 网络策略

使用以下 **NetworkPolicy** 来允许外部流量，而不考虑路由器配置：

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-from-router
spec:
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          policy-group.network.openshift.io/ingress: "" 1
  podSelector: {}
  policyTypes:
  - Ingress
```

1. **policy-group.network.openshift.io/ingress:""** 标签支持 OVN-Kubernetes。

3.1.1.2. 使用 allow-from-hostnetwork 网络策略

添加以下 **allow-from-hostnetwork NetworkPolicy** 对象来指示来自主机网络 pod 的流量。

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-from-hostnetwork
spec:
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          policy-group.network.openshift.io/host-network: ""
  podSelector: {}
  policyTypes:
  - Ingress
```

3.1.2. 使用 OVN-Kubernetes 网络插件优化网络策略

在设计您的网络策略时，请参考以下指南：

- 对于具有相同 **spec.podSelector** spec 的网络策略，使用带有多个 **ingress** 或 **egress** 规则的一个网络策略比带有 **ingress** 或 **egress** 子集的多个网络策略更高效。
- 每个基于 **podSelector** 或 **namespaceSelector** spec 的 **ingress** 或 **egress** 规则会生成一个的 OVS 流数量，它与**由网络策略选择的 pod 数量 + 由 ingress 或 egress 选择的 pod 数量**成比例因此，最好使用在一个规则中可以选择您所需的 pod 的 **podSelector** 或 **namespaceSelector** 规格，而不是为每个 pod 创建单独的规则。
  例如，以下策略包含两个规则：

  ```
  apiVersion: networking.k8s.io/v1
  kind: NetworkPolicy
  metadata:
    name: test-network-policy
  spec:
    podSelector: {}
    ingress:
    - from:
      - podSelector:
          matchLabels:
            role: frontend
    - from:
      - podSelector:
          matchLabels:
            role: backend
  ```

  以下策略表示这两个规则与以下相同的规则：

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

```
  - podSelector:
      matchExpressions:
      - {key: role, operator: In, values: [frontend, backend]}
```

相同的指南信息适用于 **spec.podSelector** spec。如果不同的网络策略有相同的 **ingress** 或 **egress** 规则，则创建一个带有通用的 **spec.podSelector** spec 可能更有效率。例如，以下两个策略有不同的规则：

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: policy1
spec:
  podSelector:
    matchLabels:
      role: db
  ingress:
  - from:
    - podSelector:
        matchLabels:
          role: frontend
---
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: policy2
spec:
  podSelector:
    matchLabels:
      role: client
  ingress:
  - from:
    - podSelector:
        matchLabels:
          role: frontend
```

以下网络策略将这两个相同的规则作为一个：

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: policy3
spec:
  podSelector:
    matchExpressions:
    - {key: role, operator: In, values: [db, client]}
  ingress:
  - from:
    - podSelector:
        matchLabels:
          role: frontend
```

当只有多个选择器表示为一个选择器时，您可以应用此优化。如果选择器基于不同的标签，则可能无法应用此优化。在这些情况下，请考虑为网络策略优化应用一些新标签。

3.1.2.1. OVN-Kubernetes 中的 NetworkPolicy CR 和外部 IP

在 OVN-Kubernetes 中，**NetworkPolicy** 自定义资源(CR)强制执行严格的隔离规则。如果服务使用外部 IP 公开，网络策略可以阻止来自其他命名空间的访问，除非明确配置为允许流量。

要允许在命名空间间访问外部 IP，请创建一个 **NetworkPolicy** CR，该 CR 明确允许来自所需命名空间的入口流量，并确保允许流量在指定的服务端口中。在不允许流量到所需端口的情况下，访问可能仍然会被限制。

输出示例

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  annotations:
  name: <policy_name>
  namespace: openshift-ingress
spec:
  ingress:
  - ports:
    - port: 80
      protocol: TCP
  - ports:
    - port: 443
      protocol: TCP
  - from:
    - namespaceSelector:
        matchLabels:
        kubernetes.io/metadata.name: <my_namespace>
  podSelector: {}
  policyTypes:
  - Ingress
```

其中：

<policy_name>
指定策略的名称。

<my_namespace>
指定部署策略的命名空间的名称。

如需了解更多详细信息，请参阅"关于网络策略"。

3.1.3. 后续步骤

- 创建网络策略
- 可选：为项目定义默认网络策略

3.1.4. 其他资源

- 项目和命名空间
- 使用网络策略配置多租户隔离

- 网络策略 API

3.2. 创建网络策略

作为集群管理员，您可以为命名空间创建网络策略。

3.2.1. 示例 NetworkPolicy 对象

以下配置注解了一个 NetworkPolicy 对象：

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: allow-27107
spec:
  podSelector:
    matchLabels:
      app: mongodb
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: app
    ports:
    - protocol: TCP
      port: 27017
```

其中：

name

NetworkPolicy 对象的名称。

spec.podSelector

描述策略应用到的 pod 的选择器。策略对象只能选择定义 NetworkPolicy 对象的项目中的 pod。

ingress.from.podSelector

与策略对象允许从中入口流量的 pod 匹配的选择器。选择器与 NetworkPolicy 在同一命名空间中的 pod 匹配。

ingress.ports

接受流量的一个或多个目标端口的列表。

3.2.2. 使用 CLI 创建网络策略

要定义细致的规则来描述集群中命名空间允许的入口或出口网络流量，您可以创建一个网络策略。

> **注意**
>
> 如果使用具有 **cluster-admin** 角色的用户登录，则可以在集群中的任何命名空间中创建网络策略。

先决条件

- 集群使用支持 **NetworkPolicy** 对象的网络插件，如带有设置了 **mode: NetworkPolicy** 的 OpenShift SDN 网络插件。
- 已安装 OpenShift CLI（**oc**）。
- 您可以使用具有 **admin** 权限的用户登陆到集群。
- 您在网络策略要应用到的命名空间中。

流程

1. 创建策略规则。

   a. 创建一个 **<policy_name>.yaml** 文件：

   ```
   $ touch <policy_name>.yaml
   ```

   其中：

   <policy_name>
   指定网络策略文件名。

   b. 在创建的文件中定义网络策略。以下示例拒绝来自所有命名空间中的所有 pod 的入口流量。这是一个基本的策略，阻止配置其他网络策略所允许的跨 pod 流量以外的所有跨 pod 网络。

   ```
   kind: NetworkPolicy
   apiVersion: networking.k8s.io/v1
   spec:
     podSelector: {}
     policyTypes:
     - Ingress
     ingress: []
   ```

   以下示例配置允许来自同一命名空间中的所有 pod 的入口流量：

   ```
   kind: NetworkPolicy
   apiVersion: networking.k8s.io/v1
   metadata:
     name: allow-same-namespace
   spec:
     podSelector:
     ingress:
     - from:
       - podSelector: {}
   # ...
   ```

   以下示例允许从一个特定命名空间到一个 pod 的入口流量。此策略允许流量从在 **namespace-y** 中运行的 pod 中获取 **pod-a** 标签。

   ```
   kind: NetworkPolicy
   apiVersion: networking.k8s.io/v1
   metadata:
     name: allow-traffic-pod
   spec:
     podSelector:
   ```

```
    matchLabels:
      pod: pod-a
  policyTypes:
  - Ingress
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          kubernetes.io/metadata.name: namespace-y
# ...
```

以下示例配置限制了服务的流量。应用此策略可确保每个带有标签 **app=bookstore** 和 **role=api** 的 Pod 只能被带有标签 **app=bookstore** 的 Pod 访问。在本例中，应用可以是 REST API 服务器，标记为标签 **app=bookstore** 和 **role=api**。

这个示例配置示例解决了以下用例：

- 将到一个服务的流量限制为仅使用需要它的其他微服务。
- 将连接限制为只允许使用它的应用程序。

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: api-allow
spec:
  podSelector:
    matchLabels:
      app: bookstore
      role: api
  ingress:
  - from:
      - podSelector:
          matchLabels:
            app: bookstore
# ...
```

2. 运行以下命令来创建网络策略对象。成功输出列出了策略对象的名称，以及 **created** 状态。

```
$ oc apply -f <policy_name>.yaml -n <namespace>
```

其中：

<policy_name>
指定网络策略文件名。

<namespace>
可选参数。如果您在与当前命名空间不同的命名空间中定义了对象，则参数会特定于命名空间。

成功输出列出了策略对象的名称，以及 **created** 状态。

注意

如果您使用 **cluster-admin** 权限登录到 web 控制台，您可以选择在集群中的任何命名空间中以 YAML 或 web 控制台的形式创建网络策略。

3.2.3. 创建默认拒绝所有网络策略

默认拒绝所有网络策略会阻止由配置其他部署的网络策略和主机网络 pod 间的网络流量允许的所有跨 pod 网络。此流程通过在 **my-project** 命名空间中应用 **deny-by-default** 策略来强制实施强大的拒绝策略。

警告

如果没有配置允许流量通信的 **NetworkPolicy** 自定义资源(CR)，以下策略可能会导致集群中的通信问题。

先决条件

- 集群使用支持 **NetworkPolicy** 对象的网络插件，如带有设置了 **mode: NetworkPolicy** 的 OpenShift SDN 网络插件。
- 已安装 OpenShift CLI（**oc**）。
- 您可以使用具有 **admin** 权限的用户登陆到集群。
- 您在网络策略要应用到的命名空间中。

流程

1. 创建以下 YAML，以定义 **deny-by-default** 策略，以拒绝所有命名空间中的所有 pod 的入口流量。将 YAML 保存到 **deny-by-default.yaml** 文件中：

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: deny-by-default
  namespace: my-project
spec:
  podSelector: {}
  ingress: []
```

其中：

namespace
: 指定要部署策略的命名空间。例如，**my-project** 命名空间。

podSelector
: 如果此字段为空，则配置与所有 pod 匹配。因此，该策略适用于 **my-project** 命名空间中的所有 pod。

ingress

其中 **[]** 表示没有指定 **ingress** 规则。这会导致传入的流量丢弃至所有 pod。

2. 输入以下命令应用策略。成功输出列出了策略对象的名称，以及 **created** 状态。

```
$ oc apply -f deny-by-default.yaml
```

3.2.4. 创建网络策略以允许来自外部客户端的流量

使用 **deny-by-default** 策略，您可以继续配置策略，允许从外部客户端到带有标签 **app=web** 的 pod 的流量。

注意

如果使用具有 **cluster-admin** 角色的用户登录，则可以在集群中的任何命名空间中创建网络策略。

按照以下步骤配置策略，以直接从公共互联网允许外部服务，或使用 Load Balancer 访问 pod。只有具有标签 **app=web** 的 pod 才允许流量。

先决条件

- 集群使用支持 **NetworkPolicy** 对象的网络插件，如带有设置了 **mode: NetworkPolicy** 的 OpenShift SDN 网络插件。
- 已安装 OpenShift CLI（**oc**）。
- 您可以使用具有 **admin** 权限的用户登陆到集群。
- 您在网络策略要应用到的命名空间中。

流程

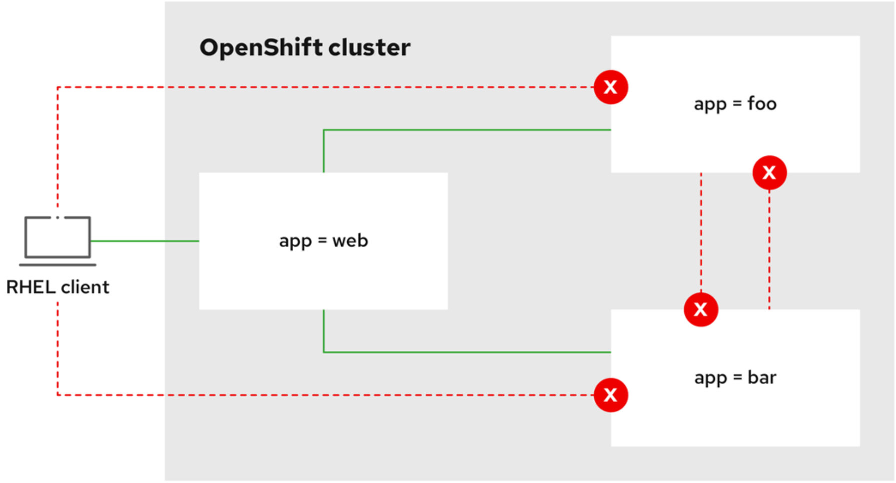
1. 创建策略，以直接从公共互联网的流量或使用负载均衡器访问 pod。将 YAML 保存到 **web-allow-external.yaml** 文件中：

```yaml
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
spec:
  policyTypes:
  - Ingress
  podSelector:
    matchLabels:
      app: web
  ingress:
    - {}
```

2. 输入以下命令应用策略。成功输出列出了策略对象的名称，以及 **created** 状态。

```
$ oc apply -f web-allow-external.yaml
```

此策略允许来自所有资源的流量，包括下图所示的外部流量：



3.2.5. 创建网络策略，允许从所有命名空间中到应用程序的流量

您可以配置一个策略，允许从所有命名空间中的所有 pod 流量到特定应用。

> **注意**
>
> 如果使用具有 **cluster-admin** 角色的用户登录，则可以在集群中的任何命名空间中创建网络策略。

先决条件

- 集群使用支持 **NetworkPolicy** 对象的网络插件，如带有设置了 **mode: NetworkPolicy** 的 OpenShift SDN 网络插件。
- 已安装 OpenShift CLI（**oc**）。
- 您可以使用具有 **admin** 权限的用户登陆到集群。
- 您在网络策略要应用到的命名空间中。

流程

1. 创建一个策略，允许从所有命名空间中的所有 pod 流量到特定应用。将 YAML 保存到 **web-allow-all-namespaces.yaml** 文件中：

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
spec:
  podSelector:
    matchLabels:
      app: web
  policyTypes:
  - Ingress
  ingress:
  - from:
    - namespaceSelector: {}
```

其中：

app

仅将策略应用到 default 命名空间中的 **app:web** pod。

namespaceSelector

选择所有命名空间中的所有 pod。

注意

默认情况下，如果您没有在策略对象中指定 **namespaceSelector** 参数，则不会选择命名空间。这意味着策略只允许从网络策略部署的命名空间的流量。

2. 输入以下命令应用策略。成功输出列出了策略对象的名称，以及 **created** 状态。

```
$ oc apply -f web-allow-all-namespaces.yaml
```

验证

1. 输入以下命令在 **default** 命名空间中启动 web 服务：

```
$ oc run web --namespace=default --image=nginx --labels="app=web" --expose --port=80
```

2. 运行以下命令在 **secondary** 命名空间中部署 **alpine** 镜像并启动 shell：

```
$ oc run test-$RANDOM --namespace=secondary --rm -i -t --image=alpine -- sh
```

3. 在 shell 中运行以下命令，并观察该服务是否允许请求：

```
# wget -qO- --timeout=2 http://web.default
```

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

3.2.6. 创建网络策略，允许从一个命名空间中到应用程序的流量

您可以配置一个允许从特定命名空间中到带有 **app=web** 标签的 pod 的策略。这个配置在以下情况下很有用：

- 将流量限制为部署了生产工作负载的命名空间。
- 启用部署到特定命名空间的监控工具，以从当前命名空间中提取指标。

注意

如果使用具有 **cluster-admin** 角色的用户登录，则可以在集群中的任何命名空间中创建网络策略。

先决条件

- 集群使用支持 **NetworkPolicy** 对象的网络插件，如带有设置了 **mode: NetworkPolicy** 的 OpenShift SDN 网络插件。
- 已安装 OpenShift CLI（**oc**）。
- 您可以使用具有 **admin** 权限的用户登陆到集群。
- 您在网络策略要应用到的命名空间中。

流程

1. 创建一个策略，允许来自特定命名空间中所有 pod 的流量，其标签为 **purpose=production**。将 YAML 保存到 **web-allow-prod.yaml** 文件中：

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: web-allow-prod
  namespace: default
spec:
  podSelector:
    matchLabels:
      app: web
  policyTypes:
  - Ingress
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          purpose: production
```

其中：

app

仅将策略应用到 default 命名空间中的 **app:web** pod。

目的

将流量仅限制为具有标签 **purpose=production** 的命名空间中的 pod。

2. 输入以下命令应用策略。成功输出列出了策略对象的名称，以及 **created** 状态。

```
$ oc apply -f web-allow-prod.yaml
```

验证

1. 输入以下命令在 **default** 命名空间中启动 web 服务：

```
$ oc run web --namespace=default --image=nginx --labels="app=web" --expose --port=80
```

2. 运行以下命令来创建 **prod** 命名空间：

```
$ oc create namespace prod
```

3. 运行以下命令来标记 **prod** 命名空间：

```
$ oc label namespace/prod purpose=production
```

4. 运行以下命令来创建 **dev** 命名空间：

```
$ oc create namespace dev
```

5. 运行以下命令来标记 **dev** 命名空间：

```
$ oc label namespace/dev purpose=testing
```

6. 运行以下命令在 **dev** 命名空间中部署 **alpine** 镜像并启动 shell：

```
$ oc run test-$RANDOM --namespace=dev --rm -i -t --image=alpine -- sh
```

7. 在 shell 中运行以下命令，并观察请求的原因。例如，预期的输出状态为 **wget: download timed out**。

```
# wget -qO- --timeout=2 http://web.default
```

8. 运行以下命令，在 **prod** 命名空间中部署 **alpine** 镜像并启动 shell：

```
$ oc run test-$RANDOM --namespace=prod --rm -i -t --image=alpine -- sh
```

9. 在 shell 中运行以下命令，并观察是否允许请求：

```
# wget -qO- --timeout=2 http://web.default
```

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
```

```
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

3.2.7. 其他资源

- 访问Web控制台
- 出口防火墙和网络策略规则的日志记录

3.3. 查看网络策略

作为集群管理员,您可以查看命名空间的网络策略。

3.3.1. 示例 NetworkPolicy 对象

以下配置注解了一个 NetworkPolicy 对象：

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: allow-27107
spec:
  podSelector:
    matchLabels:
      app: mongodb
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: app
    ports:
    - protocol: TCP
      port: 27017
```

其中：

name

NetworkPolicy 对象的名称。

spec.podSelector

描述策略应用到的 pod 的选择器。策略对象只能选择定义 NetworkPolicy 对象的项目中的 pod。

ingress.from.podSelector

与策略对象允许从中入口流量的 pod 匹配的选择器。选择器与 NetworkPolicy 在同一命名空间中的 pod 匹配。

ingress.ports

接受流量的一个或多个目标端口的列表。

3.3.2. 使用 CLI 查看网络策略

您可以检查命名空间中的网络策略。

注意

如果使用具有 **cluster-admin** 角色的用户登录，您可以查看集群中的任何网络策略。

前提条件

- 已安装 OpenShift CLI（**oc**）。
- 您可以使用具有 **admin** 权限的用户登陆到集群。
- 您在网络策略所在的命名空间中。

流程

1. 列出命名空间中的网络策略。
   a. 要查看命名空间中定义的网络策略对象，请输入以下命令：

   ```
   $ oc get networkpolicy
   ```

   b. 可选：要检查特定的网络策略，请输入以下命令：

   ```
   $ oc describe networkpolicy <policy_name> -n <namespace>
   ```

   其中：

   <policy_name>

   指定要检查的网络策略的名称。

   <namespace>

   可选：如果对象在与当前命名空间不同的命名空间中定义，使用它来指定命名空间。

   ```
   $ oc describe networkpolicy allow-same-namespace
   ```

   ```
   Name:         allow-same-namespace
   Namespace:    ns1
   Created on:   2021-05-24 22:28:56 -0400 EDT
   Labels:       <none>
   Annotations:  <none>
   Spec:
     PodSelector:     <none> (Allowing the specific traffic to all pods in this namespace)
   ```

```
Allowing ingress traffic:
  To Port: <any> (traffic allowed to all ports)
  From:
    PodSelector: <none>
Not affecting egress traffic
Policy Types: Ingress
```

注意

如果您使用 **cluster-admin** 权限登录到 web 控制台，您可以选择在集群中的任何命名空间中以 YAML 或 web 控制台的形式查看网络策略。

3.4. 编辑网络策略

作为集群管理员，您可以编辑命名空间的现有网络策略。

3.4.1. 编辑网络策略

您可以编辑命名空间中的网络策略。

注意

如果使用具有 **cluster-admin** 角色的用户登录，则可以在集群中的任何命名空间中编辑网络策略。

先决条件

- 集群使用支持 **NetworkPolicy** 对象的网络插件，如带有设置了 **mode: NetworkPolicy** 的 OpenShift SDN 网络插件。
- 已安装 OpenShift CLI（**oc**）。
- 您可以使用具有 **admin** 权限的用户登陆到集群。
- 您在网络策略所在的命名空间中。

流程

1. 可选：要列出一个命名空间中的网络策略对象，请输入以下命令：

```
$ oc get network policy -n <namespace>
```

其中：

<namespace>
可选：如果对象在与当前命名空间不同的命名空间中定义，使用它来指定命名空间。

2. 编辑网络策略对象。

   a. 如果您在文件中保存了网络策略定义，请编辑该文件并进行必要的更改，然后输入以下命令。

```
$ oc apply -n <namespace> -f <policy_file>.yaml
```

其中：

<namespace>
可选： 如果对象在与当前命名空间不同的命名空间中定义，使用它来指定命名空间。

<policy_file>
指定包含网络策略的文件的名称。

b. 如果您需要直接更新网络策略对象，请输入以下命令：

```
$ oc edit network policy <policy_name> -n <namespace>
```

其中：

<policy_name>
指定网络策略的名称。

<namespace>
可选： 如果对象在与当前命名空间不同的命名空间中定义，使用它来指定命名空间。

3. 确认网络策略对象已更新。

```
$ oc describe networkpolicy <policy_name> -n <namespace>
```

其中：

<policy_name>
指定网络策略的名称。

<namespace>
可选： 如果对象在与当前命名空间不同的命名空间中定义，使用它来指定命名空间。

注意

如果您使用 **cluster-admin** 权限登录到 web 控制台，您可以选择在集群中的任何命名空间中以 YAML 或通过 **Actions** 菜单从 web 控制台中的策略编辑网络策略。

3.4.2. 示例 NetworkPolicy 对象

以下配置注解了一个 NetworkPolicy 对象：

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: allow-27107
spec:
  podSelector:
    matchLabels:
      app: mongodb
  ingress:
  - from:
    - podSelector:
        matchLabels:
```

```
      app: app
  ports:
  - protocol: TCP
    port: 27017
```

其中：

name

NetworkPolicy 对象的名称。

spec.podSelector

描述策略应用到的 pod 的选择器。策略对象只能选择定义 NetworkPolicy 对象的项目中的 pod。

ingress.from.podSelector

与策略对象允许从中入口流量的 pod 匹配的选择器。选择器与 NetworkPolicy 在同一命名空间中的 pod 匹配。

ingress.ports

接受流量的一个或多个目标端口的列表。

3.4.3. 其他资源

- 创建网络策略

3.5. 删除网络策略

作为集群管理员，您可以从命名空间中删除网络策略。

3.5.1. 使用 CLI 删除网络策略

您可以删除命名空间中的网络策略。

注意

如果使用具有 **cluster-admin** 角色的用户登录，您可以删除集群中的任何网络策略。

先决条件

- 集群使用支持 **NetworkPolicy** 对象的网络插件，如带有设置了 **mode: NetworkPolicy** 的 OpenShift SDN 网络插件。
- 已安装 OpenShift CLI（**oc**）。
- 您可以使用具有 **admin** 权限的用户登陆到集群。
- 您在网络策略所在的命名空间中。

流程

- 要删除网络策略对象，请输入以下命令。成功输出列出了策略对象的名称，以及 **deleted** 状态。

```
$ oc delete networkpolicy <policy_name> -n <namespace>
```

其中：

<policy_name>

指定网络策略的名称。

<namespace>

可选参数。如果您在与当前命名空间不同的命名空间中定义了对象，则参数会特定于命名空间。

注意

如果使用 **cluster-admin** 权限登录到 web 控制台，您可以选择在集群上以 YAML 或通过 **Actions** 菜单从 web 控制台中的策略删除网络策略。

3.6. 为项目定义默认网络策略

作为集群管理员，您可以在创建新项目时修改新项目模板，使其自动包含网络策略。如果您还没有新项目的自定义模板，则需要首先创建一个。

3.6.1. 为新项目修改模板

作为集群管理员，您可以修改默认项目模板，以便使用自定义要求创建新项目。

创建自己的自定义项目模板：

先决条件

- 可以使用具有 **cluster-admin** 权限的账户访问 OpenShift Container Platform 集群。

流程

1. 以具有 **cluster-admin** 特权的用户身份登录。
2. 生成默认项目模板：

```
$ oc adm create-bootstrap-project-template -o yaml > template.yaml
```

3. 使用文本编辑器，通过添加对象或修改现有对象来修改生成的 **template.yaml** 文件。
4. 项目模板必须创建在 **openshift-config** 命名空间中。加载修改后的模板：

```
$ oc create -f template.yaml -n openshift-config
```

5. 使用 Web 控制台或 CLI 编辑项目配置资源。
   - 使用 Web 控制台：
     i. 导航至 **Administration → Cluster Settings** 页面。
     ii. 单击 **Configuration** 以查看所有配置资源。
     iii. 找到 **Project** 的条目，并点击 **Edit YAML**。
   - 使用 CLI：
     i. 编辑 **project.config.openshift.io/cluster** 资源：

```
$ oc edit project.config.openshift.io/cluster
```

6. 更新 **spec** 部分，使其包含 **projectRequestTemplate** 和 **name** 参数，再设置您上传的项目模板的名称。默认名称为 **project-request**。

带有自定义项目模板的项目配置资源

```
apiVersion: config.openshift.io/v1
kind: Project
metadata:
# ...
spec:
  projectRequestTemplate:
    name: <template_name>
# ...
```

7. 保存更改后，创建一个新项目来验证是否成功应用了您的更改。

3.6.2. 在新项目模板中添加网络策略

作为集群管理员，您可以在新项目的默认模板中添加网络策略。OpenShift Container Platform 将自动创建项目中模板中指定的所有 **NetworkPolicy** 对象。

先决条件

- 集群使用支持 **NetworkPolicy** 对象的默认容器网络接口(CNI)网络插件，如 OVN-Kubernetes。
- 已安装 OpenShift CLI（**oc**）。
- 您需要使用具有 **cluster-admin** 权限的用户登陆到集群。
- 您必须已为新项目创建了自定义的默认项目模板。

流程

1. 运行以下命令来编辑新项目的默认模板：

```
$ oc edit template <project_template> -n openshift-config
```

将 **<project_template>** 替换为您为集群配置的缺省模板的名称。默认模板名称为 **project-request**。

2. 在模板中，将每个 **NetworkPolicy** 对象作为一个元素添加到 **objects** 参数中。**objects** 参数可以是一个或多个对象的集合。
在以下示例中，**objects** 参数集合包括几个 **NetworkPolicy** 对象。

```
objects:
- apiVersion: networking.k8s.io/v1
  kind: NetworkPolicy
  metadata:
    name: allow-from-same-namespace
  spec:
    podSelector: {}
```

```
    ingress:
    - from:
      - podSelector: {}
- apiVersion: networking.k8s.io/v1
  kind: NetworkPolicy
  metadata:
    name: allow-from-openshift-ingress
  spec:
    ingress:
    - from:
      - namespaceSelector:
          matchLabels:
            policy-group.network.openshift.io/ingress:
    podSelector: {}
    policyTypes:
    - Ingress
- apiVersion: networking.k8s.io/v1
  kind: NetworkPolicy
  metadata:
    name: allow-from-kube-apiserver-operator
  spec:
    ingress:
    - from:
      - namespaceSelector:
          matchLabels:
            kubernetes.io/metadata.name: openshift-kube-apiserver-operator
        podSelector:
          matchLabels:
            app: kube-apiserver-operator
    policyTypes:
    - Ingress
...
```

3. 可选：创建一个新项目，并确认您的网络策略对象成功创建。

   a. 创建一个新项目

   ```
   $ oc new-project <project> 1
   ```

   1 将 **<project>** 替换为您要创建的项目的名称。

   b. 确认新项目模板中的网络策略对象存在于新项目中：

   ```
   $ oc get networkpolicy
   ```

   预期输出：

   ```
   NAME                        POD-SELECTOR   AGE
   allow-from-openshift-ingress   <none>         7s
   allow-from-same-namespace      <none>         7s
   ```

3.7. 使用网络策略配置多租户隔离

作为集群管理员，您可以配置网络策略以为多租户网络提供隔离功能。

> **注意**
>
> 本节所述配置网络策略提供了在以前版本的 OpenShift Container Platform 中与 OpenShift SDN 的多租户模式类似的网络隔离。

3.7.1. 使用网络策略配置多租户隔离

您可以配置项目，使其与其他项目命名空间中的 pod 和服务分离。

先决条件

- 集群使用支持 **NetworkPolicy** 对象的网络插件，如带有设置了 **mode: NetworkPolicy** 的 OpenShift SDN 网络插件。
- 已安装 OpenShift CLI（**oc**）。
- 您可以使用具有 **admin** 权限的用户登陆到集群。

流程

1. 创建以下 **NetworkPolicy** 对象：
   a. 名为 **allow-from-openshift-ingress** 的策略。

```
$ cat << EOF| oc create -f -
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-from-openshift-ingress
spec:
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          policy-group.network.openshift.io/ingress: ""
  podSelector: {}
  policyTypes:
  - Ingress
EOF
```

> **注意**
>
> **policy-group.network.openshift.io/ingress: ""** 是 OVN-Kubernetes 的首选命名空间选择器标签。

   b. 名为 **allow-from-openshift-monitoring** 的策略：

```
$ cat << EOF| oc create -f -
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-from-openshift-monitoring
```

```
spec:
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          network.openshift.io/policy-group: monitoring
  podSelector: {}
  policyTypes:
  - Ingress
EOF
```

c. 名为 **allow-same-namespace** 的策略：

```
$ cat << EOF| oc create -f -
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: allow-same-namespace
spec:
  podSelector:
  ingress:
  - from:
    - podSelector: {}
EOF
```

d. 名为 **allow-from-kube-apiserver-operator** 的策略：

```
$ cat << EOF| oc create -f -
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-from-kube-apiserver-operator
spec:
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          kubernetes.io/metadata.name: openshift-kube-apiserver-operator
      podSelector:
        matchLabels:
          app: kube-apiserver-operator
  policyTypes:
  - Ingress
EOF
```

如需了解更多详细信息，请参阅 新的**kube-apiserver-operator** Webhook 控制器验证 Webhook 的健康状况。

2. 可选： 要确认当前项目中存在网络策略，请输入以下命令：

```
$ oc describe networkpolicy
```

输出示例

```
Name:         allow-from-openshift-ingress
Namespace:    example1
Created on:   2020-06-09 00:28:17 -0400 EDT
Labels:       <none>
Annotations:  <none>
Spec:
  PodSelector:     <none> (Allowing the specific traffic to all pods in this namespace)
  Allowing ingress traffic:
    To Port: <any> (traffic allowed to all ports)
    From:
      NamespaceSelector: policy-group.network.openshift.io/ingress:
  Not affecting egress traffic
  Policy Types: Ingress


Name:         allow-from-openshift-monitoring
Namespace:    example1
Created on:   2020-06-09 00:29:57 -0400 EDT
Labels:       <none>
Annotations:  <none>
Spec:
  PodSelector:     <none> (Allowing the specific traffic to all pods in this namespace)
  Allowing ingress traffic:
    To Port: <any> (traffic allowed to all ports)
    From:
      NamespaceSelector: network.openshift.io/policy-group: monitoring
  Not affecting egress traffic
  Policy Types: Ingress
```

3.7.2. 后续步骤

- 为项目定义默认网络策略

第 4 章 网络安全的审计日志记录

OVN-Kubernetes 网络插件使用 Open Virtual Network (OVN) 访问控制列表 (ACL) 来管理 **AdminNetworkPolicy**、**BaselineAdminNetworkPolicy**、**NetworkPolicy** 和 **EgressFirewall** 对象。审计日志记录会公开 **NetworkPolicy**, **EgressFirewall** 和 **BaselineAdminNetworkPolicy** 自定义资源 (CR) 的 **allow** 和 **deny** ACL 事件。日志记录还公开 **AdminNetworkPolicy** (ANP) CR 的 **allow**, **deny**, 和 **pass** ACL 的事件。

注意

审计日志记录仅适用于 OVN-Kubernetes 网络插件。

4.1. 审计配置

审计日志记录的配置作为 OVN-Kubernetes 集群网络配置的一部分指定。以下 YAML 演示了审计日志的默认值：

审计日志记录配置

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      policyAuditConfig:
        destination: "null"
        maxFileSize: 50
        rateLimit: 20
        syslogFacility: local0
```

下表描述了审计日志的配置字段。

表 4.1. policyAuditConfig object

字段	类型	描述
rateLimit	整数	每个节点每秒生成一次的消息数量上限。默认值为每秒 **20** 条消息。
maxFileSize	整数	审计日志的最大大小，以字节为单位。默认值为 **50000000** 或 50 MB。
maxLogFiles	整数	保留的日志文件的最大数量。

字段	类型	描述
目的地	字符串	以下附加审计日志目标之一： **libc** 主机上的 journald 进程的 libc **syslog（）** 函数。 **UDP:<host>:<port>** 一个 syslog 服务器。将 **<host>:<port> 替换为 syslog 服务器的主机** 和端口。 **Unix:<file>** 由 **<file>** 指定的 Unix 域套接字文件。 **null** 不要将审计日志发送到任何其他目标。
syslogFacility	字符串	syslog 工具，如 as **kern**，如 RFC5424 定义。默认值为 **local0**。

4.2. 审计日志记录

您可以为审计日志配置目的地，如 syslog 服务器或 UNIX 域套接字。无论任何其他配置如何，审计日志始终保存到集群中的每个 OVN-Kubernetes pod 上的 **/var/log/ovn/acl-audit-log**。

您可以使用 **k8s.ovn.org/acl-logging** 部分为每个命名空间启用审计日志记录。在 **k8s.ovn.org/acl-logging** 部分中，您必须指定 **allow**、**deny** 或这两个值来为命名空间启用审计日志记录。

注意

网络策略不支持将 **Pass** 操作设置为规则。

ACL-logging 实现记录网络的访问控制列表 (ACL) 事件。您可以查看这些日志来分析任何潜在的安全问题。

命名空间注解示例

```
kind: Namespace
apiVersion: v1
metadata:
  name: example1
  annotations:
    k8s.ovn.org/acl-logging: |-
      {
        "deny": "info",
        "allow": "info"
      }
```

要查看默认的 ACL 日志记录配置值，请参阅 **cluster-network-03-config.yml** 文件中的 **policyAuditConfig** 对象。如果需要，您可以更改此文件中的日志文件参数的 ACL 日志记录配置值。

日志信息格式与 RFC5424 中定义的 syslog 兼容。syslog 工具可配置，默认为 **local0**。以下示例显示了日志消息中输出的关键参数及其值：

输出参数及其值的日志记录消息示例

```
<timestamp>|<message_serial>|acl_log(ovn_pinctrl0)|<severity>|name="<acl_name>", verdict="<verdict>", severity="<severity>", direction="<direction>": <flow>
```

其中：

- **<timestamp>** 声明创建日志消息的时间和日期。
- **<message_serial>** 列出日志消息的序列号。
- **acl_log (ovn_pinctrl0)** 是一个字面字符串，它会在 OVN-Kubernetes 插件中输出日志消息的位置。
- **<severity>** 为日志消息设置严重性级别。如果您启用支持 **allow** 和 **deny** 任务的审计日志记录，则日志消息输出中会显示两个严重性级别。
- **<name>** 说明由网络策略创建的 OVN Network Bridging Database (**nbdb**) 中的 ACL-logging 实现的名称。
- **<verdict>** 可以是 **allow** 或 **drop**。
- **<direction>** 可以是 **to-lport** 或 **from-lport**，表示策略应用到 pod 的流量。
- **<flow>** 显示与 **OpenFlow** 协议等效的格式的数据包信息。此参数包含 Open vSwitch (OVS) 字段。

以下示例显示了 **flow** 参数用来从系统内存提取数据包信息的 OVS 字段：

flow 参数用来提取数据包信息的 OVS 字段示例

```
<proto>,vlan_tci=0x0000,dl_src=<src_mac>,dl_dst=<source_mac>,nw_src=<source_ip>,nw_dst=<target_ip>,nw_tos=<tos_dscp>,nw_ecn=<tos_ecn>,nw_ttl=<ip_ttl>,nw_frag=<fragment>,tp_src=<tcp_src_port>,tp_dst=<tcp_dst_port>,tcp_flags=<tcp_flags>
```

其中：

- **<proto>** 声明协议。有效值为 **tcp** 和 **udp**。
- **vlan_tci=0x0000** 声明 VLAN 标头为 **0**，因为没有为内部 pod 网络流量设置 VLAN ID。
- **<src_mac>** 指定 Media Access Control (MAC) 地址的源。
- **<source_mac>** 指定 MAC 地址的目的地。
- **<source_ip>** 列出源 IP 地址
- **<target_ip>** 列出目标 IP 地址。
- **<tos_dscp>** 声明 Differentiated Services Code Point (DSCP) 值，对网络流量进行分类并进行优先级排序。

- **<tos_ecn>** 声明 Explicit Congestion Notification (ECN) 值，它表示您的网络中的阻塞网络流量。
- **<ip_ttl>** 声明数据包的 Time To Live (TTP) 信息。
- **<fragment>** 指定要匹配的 IP 片段或 IP 非碎片。
- **<tcp_src_port>** 显示 TCP 和 UDP 协议的端口源。
- **<tcp_dst_port>** 列出 TCP 和 UDP 协议的目的地端口。
- **<tcp_flags>** 支持多个标示，如 **SYN**, **ACK**, **PSH** 等。如果您需要设置多个值，则不同的值由竖线 (|) 分隔。UDP 协议不支持此参数。

注意

有关前面的字段描述的更多信息，请转至 **ovs-fields** 的 OVS 手册页。

网络策略的 ACL 拒绝日志条目示例

```
2023-11-02T16:28:54.139Z|00004|acl_log(ovn_pinctrl0)|INFO|name="NP:verify-audit-logging:Ingress", verdict=drop, severity=alert, direction=to-lport: tcp,vlan_tci=0x0000,dl_src=0a:58:0a:81:02:01,dl_dst=0a:58:0a:81:02:23,nw_src=10.131.0.39,nw_dst=10.129.2.35,nw_tos=0,nw_ecn=0,nw_ttl=62,nw_frag=no,tp_src=58496,tp_dst=8080,tcp_flags=syn
2023-11-02T16:28:55.187Z|00005|acl_log(ovn_pinctrl0)|INFO|name="NP:verify-audit-logging:Ingress", verdict=drop, severity=alert, direction=to-lport: tcp,vlan_tci=0x0000,dl_src=0a:58:0a:81:02:01,dl_dst=0a:58:0a:81:02:23,nw_src=10.131.0.39,nw_dst=10.129.2.35,nw_tos=0,nw_ecn=0,nw_ttl=62,nw_frag=no,tp_src=58496,tp_dst=8080,tcp_flags=syn
2023-11-02T16:28:57.235Z|00006|acl_log(ovn_pinctrl0)|INFO|name="NP:verify-audit-logging:Ingress", verdict=drop, severity=alert, direction=to-lport: tcp,vlan_tci=0x0000,dl_src=0a:58:0a:81:02:01,dl_dst=0a:58:0a:81:02:23,nw_src=10.131.0.39,nw_dst=10.129.2.35,nw_tos=0,nw_ecn=0,nw_ttl=62,nw_frag=no,tp_src=58496,tp_dst=8080,tcp_flags=syn
```

下表描述了命名空间注解值：

表 4.2. k8s.ovn.org/acl-logging 的审计日志记录命名空间注解

字段	描述
deny	阻止任何与 **deny** 操作匹配的 ACL 规则的流量的访问。字段支持 **alert**, **warning**, **notice**, **info**, 或 **debug** 值。
allow	允许命名空间访问与 **allow** 操作匹配的 ACL 规则的任何流量。字段支持 **alert**, **warning**, **notice**, **info**, 或 **debug** 值。
pass	**pass** 操作适用于管理员网络策略的 ACL 规则。**pass** 操作允许命名空间中的网络策略或基准 admin 网络策略规则来评估所有传入和传出流量。网络策略不支持 **pass** 操作。

其他资源

- 了解网络策略 API

4.3. ADMINNETWORKPOLICY 审计日志记录

每个 **AdminNetworkPolicy** CR 启用了审计日志记录，使用 **k8s.ovn.org/acl-logging** 键注解 ANP 策略，如下例所示：

例 4.1. AdminNetworkPolicy CR 的注解示例

```
apiVersion: policy.networking.k8s.io/v1alpha1
kind: AdminNetworkPolicy
metadata:
  annotations:
    k8s.ovn.org/acl-logging: '{ "deny": "alert", "allow": "alert", "pass" : "warning" }'
  name: anp-tenant-log
spec:
  priority: 5
  subject:
    namespaces:
      matchLabels:
        tenant: backend-storage # Selects all pods owned by storage tenant.
  ingress:
    - name: "allow-all-ingress-product-development-and-customer" # Product development and customer tenant ingress to backend storage.
      action: "Allow"
      from:
      - pods:
          namespaceSelector:
            matchExpressions:
            - key: tenant
              operator: In
              values:
              - product-development
              - customer
          podSelector: {}
    - name: "pass-all-ingress-product-security"
      action: "Pass"
      from:
      - namespaces:
          matchLabels:
              tenant: product-security
    - name: "deny-all-ingress" # Ingress to backend from all other pods in the cluster.
      action: "Deny"
      from:
      - namespaces: {}
  egress:
    - name: "allow-all-egress-product-development"
      action: "Allow"
      to:
      - pods:
          namespaceSelector:
            matchLabels:
              tenant: product-development
          podSelector: {}
    - name: "pass-egress-product-security"
      action: "Pass"
      to:
      - namespaces:
```

```
      matchLabels:
        tenant: product-security
  - name: "deny-all-egress" # Egress from backend denied to all other pods.
    action: "Deny"
    to:
    - namespaces: {}
```

每当特定的 OVN ACL 达到并满足日志记录注解中设置的操作条件时，都会生成日志。例如，一个事件，其中任何带有标签 **tenant: product-development** 的命名空间都访问带有标签 **tenant: backend-storage** 的命名空间，则会生成日志。

注意

ACL 日志记录限制为 60 个字符。如果您的 ANP **name** 字段较长，日志的其余部分将被截断。

以下是以下示例日志条目的方向索引：

方向	规则
入口	**Rule0** 允许从租户 **product-development** 和 **customer** 到租户 **backend-storage**; Ingress0: **Allow** **Rule1** 从 **product-security`** 到租户 **`backend-storage**; Ingress1: **Pass** 传递 **Rule2** 拒绝来自所有 pod 的入站流量; Ingress2: **Deny**
Egress	**Rule0** 允许到 **product-development**; Egress0: **Allow** **Rule1** 传递给 **product-security**; Egress1 **Pass** **Rule2** 拒绝到所有其他 pod 的出口流量; Egress2: **Deny**

例 4.2. ACL 日志项示例，用于带有 Ingress:0 和 Egress:0 的名为 anp-tenant-log 的 AdminNetworkPolicy 的 Allow 操作

```
2024-06-10T16:27:45.194Z|00052|acl_log(ovn_pinctrl0)|INFO|name="ANP:anp-tenant-log:Ingress:0", verdict=allow, severity=alert, direction=to-lport: tcp,vlan_tci=0x0000,dl_src=0a:58:0a:80:02:1a,dl_dst=0a:58:0a:80:02:19,nw_src=10.128.2.26,nw_dst=10.128.2.25,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=57814,tp_dst=8080,tcp_flags=syn

2024-06-10T16:28:23.130Z|00059|acl_log(ovn_pinctrl0)|INFO|name="ANP:anp-tenant-log:Ingress:0", verdict=allow, severity=alert, direction=to-lport: tcp,vlan_tci=0x0000,dl_src=0a:58:0a:80:02:18,dl_dst=0a:58:0a:80:02:19,nw_src=10.128.2.24,nw_ds
```

```
t=10.128.2.25,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=38620,tp_dst=8080,tcp_flags=ack

2024-06-10T16:28:38.293Z|00069|acl_log(ovn_pinctrl0)|INFO|name="ANP:anp-tenant-
log:Egress:0", verdict=allow, severity=alert, direction=from-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:80:02:19,dl_dst=0a:58:0a:80:02:1a,nw_src=10.128.2.25,nw_ds
t=10.128.2.26,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=47566,tp_dst=8080,tcp_flags=fin|a
ck=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=55704,tp_dst=8080,tcp_flags=ack
```

例 4.3. ACL 日志项示例，用于带有 Ingress:1 和 Egress:1 的名为 anp-tenant-log 的 AdminNetworkPolicy 的 Pass 操作

```
2024-06-10T16:33:12.019Z|00075|acl_log(ovn_pinctrl0)|INFO|name="ANP:anp-tenant-
log:Ingress:1", verdict=pass, severity=warning, direction=to-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:80:02:1b,dl_dst=0a:58:0a:80:02:19,nw_src=10.128.2.27,nw_ds
t=10.128.2.25,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=37394,tp_dst=8080,tcp_flags=ack

2024-06-10T16:35:04.209Z|00081|acl_log(ovn_pinctrl0)|INFO|name="ANP:anp-tenant-
log:Egress:1", verdict=pass, severity=warning, direction=from-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:80:02:19,dl_dst=0a:58:0a:80:02:1b,nw_src=10.128.2.25,nw_ds
t=10.128.2.27,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=34018,tp_dst=8080,tcp_flags=ack
```

例 4.4. ACL 日志项示例，用于带有 Ingress:2 和 Egress:2 的名为 anp-tenant-log 的 AdminNetworkPolicy 的 Deny 操作

```
2024-06-10T16:43:05.287Z|00087|acl_log(ovn_pinctrl0)|INFO|name="ANP:anp-tenant-
log:Egress:2", verdict=drop, severity=alert, direction=from-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:80:02:19,dl_dst=0a:58:0a:80:02:18,nw_src=10.128.2.25,nw_ds
t=10.128.2.24,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=51598,tp_dst=8080,tcp_flags=syn

2024-06-10T16:44:43.591Z|00090|acl_log(ovn_pinctrl0)|INFO|name="ANP:anp-tenant-
log:Ingress:2", verdict=drop, severity=alert, direction=to-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:80:02:1c,dl_dst=0a:58:0a:80:02:19,nw_src=10.128.2.28,nw_ds
=10.128.2.25,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=33774,tp_dst=8080,tcp_flags=syn
```

下表描述了 ANP 注解：

表 4.3. 审计日志记录 AdminNetworkPolicy 注解

注解	value

注解	value
k8s.ovn.org/acl-logging	您必须至少指定 **Allow**、**Deny** 或 **Pass** 之一才能为命名空间启用审计日志记录。 **Deny** 可选：指定 **alert**、**warning**、**notice**、**info** 或 **debug**。 **Allow** 可选：指定 **alert**、**warning**、**notice**、**info** 或 **debug**。 **Pass** 可选：指定 **alert**、**warning**、**notice**、**info** 或 **debug**。

4.4. BASELINEADMINNETWORKPOLICY 审计日志记录

使用 **k8s.ovn.org/acl-logging** 键注解 BANP 策略，在 **BaselineAdminNetworkPolicy** CR 中启用审计日志记录，如下例所示：

例 4.5. BaselineAdminNetworkPolicy CR 的注解示例

```
apiVersion: policy.networking.k8s.io/v1alpha1
kind: BaselineAdminNetworkPolicy
metadata:
  annotations:
    k8s.ovn.org/acl-logging: '{ "deny": "alert", "allow": "alert"}'
  name: default
spec:
  subject:
    namespaces:
      matchLabels:
          tenant: workloads # Selects all workload pods in the cluster.
  ingress:
  - name: "default-allow-dns" # This rule allows ingress from dns tenant to all workloads.
    action: "Allow"
    from:
    - namespaces:
        matchLabels:
          tenant: dns
  - name: "default-deny-dns" # This rule denies all ingress from all pods to workloads.
    action: "Deny"
    from:
    - namespaces: {} # Use the empty selector with caution because it also selects OpenShift namespaces as well.
  egress:
  - name: "default-deny-dns" # This rule denies all egress from workloads. It will be applied when no ANP or network policy matches.
    action: "Deny"
    to:
    - namespaces: {} # Use the empty selector with caution because it also selects OpenShift namespaces as well.
```

在示例中，一个事件，其中任何带有标签 **tenant: dns** 的命名空间都访问带有标签 **tenant: workload** 的命名空间，则会生成日志。

以下是以下示例日志条目的方向索引：

方向	规则
入口	**Rule0** 允许从租户 **dns** 到租户 **workloads**; Ingress0: **Allow** **Rule1** 拒绝从所有 pod 到租户 **workloads**; Ingress1: **Deny**
Egress	**Rule0** 拒绝所有 pod; Egress0: **Deny**

例 4.6. ACL allow 日志项示例，用于带有Ingress:0 的 default BANP 的Allow 操作

```
2024-06-10T18:11:58.263Z|00022|acl_log(ovn_pinctrl0)|INFO|name="BANP:default:Ingress:0",
verdict=allow, severity=alert, direction=to-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:82:02:57,dl_dst=0a:58:0a:82:02:56,nw_src=10.130.2.87,nw_ds
t=10.130.2.86,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=60510,tp_dst=8080,tcp_flags=syn

2024-06-10T18:11:58.264Z|00023|acl_log(ovn_pinctrl0)|INFO|name="BANP:default:Ingress:0",
verdict=allow, severity=alert, direction=to-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:82:02:57,dl_dst=0a:58:0a:82:02:56,nw_src=10.130.2.87,nw_ds
t=10.130.2.86,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=60510,tp_dst=8080,tcp_flags=psh
ack
2024-06-10T18:11:58.264Z|00024|acl_log(ovn_pinctrl0)|INFO|name="BANP:default:Ingress:0",
verdict=allow, severity=alert, direction=to-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:82:02:57,dl_dst=0a:58:0a:82:02:56,nw_src=10.130.2.87,nw_ds
t=10.130.2.86,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=60510,tp_dst=8080,tcp_flags=ack

2024-06-10T18:11:58.264Z|00025|acl_log(ovn_pinctrl0)|INFO|name="BANP:default:Ingress:0",
verdict=allow, severity=alert, direction=to-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:82:02:57,dl_dst=0a:58:0a:82:02:56,nw_src=10.130.2.87,nw_ds
t=10.130.2.86,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=60510,tp_dst=8080,tcp_flags=ack

2024-06-10T18:11:58.264Z|00026|acl_log(ovn_pinctrl0)|INFO|name="BANP:default:Ingress:0",
verdict=allow, severity=alert, direction=to-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:82:02:57,dl_dst=0a:58:0a:82:02:56,nw_src=10.130.2.87,nw_ds
t=10.130.2.86,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=60510,tp_dst=8080,tcp_flags=fin|a
ck
2024-06-10T18:11:58.264Z|00027|acl_log(ovn_pinctrl0)|INFO|name="BANP:default:Ingress:0",
verdict=allow, severity=alert, direction=to-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:82:02:57,dl_dst=0a:58:0a:82:02:56,nw_src=10.130.2.87,nw_ds
t=10.130.2.86,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=60510,tp_dst=8080,tcp_flags=ack
```

例 4.7. ACL allow 日志项示例，用于带有Egress:0 和 Ingress:1 的 default BANP 的Allow 操作

```
2024-06-10T18:09:57.774Z|00016|acl_log(ovn_pinctrl0)|INFO|name="BANP:default:Egress:0", verdict=drop, severity=alert, direction=from-lport: tcp,vlan_tci=0x0000,dl_src=0a:58:0a:82:02:56,dl_dst=0a:58:0a:82:02:57,nw_src=10.130.2.86,nw_dst=10.130.2.87,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=45614,tp_dst=8080,tcp_flags=syn

2024-06-10T18:09:58.809Z|00017|acl_log(ovn_pinctrl0)|INFO|name="BANP:default:Egress:0", verdict=drop, severity=alert, direction=from-lport: tcp,vlan_tci=0x0000,dl_src=0a:58:0a:82:02:56,dl_dst=0a:58:0a:82:02:57,nw_src=10.130.2.86,nw_dst=10.130.2.87,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=45614,tp_dst=8080,tcp_flags=syn

2024-06-10T18:10:00.857Z|00018|acl_log(ovn_pinctrl0)|INFO|name="BANP:default:Egress:0", verdict=drop, severity=alert, direction=from-lport: tcp,vlan_tci=0x0000,dl_src=0a:58:0a:82:02:56,dl_dst=0a:58:0a:82:02:57,nw_src=10.130.2.86,nw_dst=10.130.2.87,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=45614,tp_dst=8080,tcp_flags=syn

2024-06-10T18:10:25.414Z|00019|acl_log(ovn_pinctrl0)|INFO|name="BANP:default:Ingress:1", verdict=drop, severity=alert, direction=to-lport: tcp,vlan_tci=0x0000,dl_src=0a:58:0a:82:02:58,dl_dst=0a:58:0a:82:02:56,nw_src=10.130.2.88,nw_dst=10.130.2.86,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=40630,tp_dst=8080,tcp_flags=syn

2024-06-10T18:10:26.457Z|00020|acl_log(ovn_pinctrl0)|INFO|name="BANP:default:Ingress:1", verdict=drop, severity=alert, direction=to-lport: tcp,vlan_tci=0x0000,dl_src=0a:58:0a:82:02:58,dl_dst=0a:58:0a:82:02:56,nw_src=10.130.2.88,nw_dst=10.130.2.86,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=40630,tp_dst=8080,tcp_flags=syn

2024-06-10T18:10:28.505Z|00021|acl_log(ovn_pinctrl0)|INFO|name="BANP:default:Ingress:1", verdict=drop, severity=alert, direction=to-lport: tcp,vlan_tci=0x0000,dl_src=0a:58:0a:82:02:58,dl_dst=0a:58:0a:82:02:56,nw_src=10.130.2.88,nw_dst=10.130.2.86,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=40630,tp_dst=8080,tcp_flags=syn
```

下表描述了 BANP 注解：

表 4.4. 审计日志记录 BaselineAdminNetworkPolicy 注解

注解	value
k8s.ovn.org/acl-logging	您必须至少指定 **Allow** 或 **Deny** 之一才能为命名空间启用审计日志记录。 **Deny** 可选：指定 **alert**、**warning**、**notice**、**info** 或 **debug**。 **Allow** 可选：指定 **alert**、**warning**、**notice**、**info** 或 **debug**。

4.5. 为集群配置出口防火墙和网络策略审计

作为集群管理员，您可以自定义集群的审计日志。

先决条件

- 安装 OpenShift CLI（**oc**）。

- 使用具有 **cluster-admin** 权限的用户登陆到集群。

流程

- 要自定义审计日志配置，请输入以下命令：

```
$ oc edit network.operator.openshift.io/cluster
```

提示

您还可以自定义并应用以下 YAML 来配置审计日志记录：

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      policyAuditConfig:
        destination: "null"
        maxFileSize: 50
        rateLimit: 20
        syslogFacility: local0
```

验证

1. 要创建带有网络策略的命名空间，请完成以下步骤：

   a. 创建命名空间进行验证：

   ```
   $ cat <<EOF| oc create -f -
   kind: Namespace
   apiVersion: v1
   metadata:
     name: verify-audit-logging
     annotations:
       k8s.ovn.org/acl-logging: '{ "deny": "alert", "allow": "alert" }'
   EOF
   ```

   成功输出列出带有网络策略的命名空间，状态为 **created**。

   b. 为命名空间创建网络策略：

   ```
   $ cat <<EOF| oc create -n verify-audit-logging -f -
   apiVersion: networking.k8s.io/v1
   kind: NetworkPolicy
   metadata:
     name: deny-all
   spec:
     podSelector:
       matchLabels:
     policyTypes:
     - Ingress
   ```

```
  - Egress
---
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-from-same-namespace
  namespace: verify-audit-logging
spec:
  podSelector: {}
  policyTypes:
   - Ingress
   - Egress
  ingress:
    - from:
        - podSelector: {}
  egress:
    - to:
       - namespaceSelector:
          matchLabels:
            kubernetes.io/metadata.name: verify-audit-logging
EOF
```

输出示例

```
networkpolicy.networking.k8s.io/deny-all created
networkpolicy.networking.k8s.io/allow-from-same-namespace created
```

2. 为 **default** 命名空间中的源流量创建 pod：

```
$ cat <<EOF| oc create -n default -f -
apiVersion: v1
kind: Pod
metadata:
  name: client
spec:
  containers:
    - name: client
      image: registry.access.redhat.com/rhel7/rhel-tools
      command: ["/bin/sh", "-c"]
      args:
        ["sleep inf"]
EOF
```

3. 在 **verify-audit-logging** 命名空间中创建两个 pod：

```
$ for name in client server; do
cat <<EOF| oc create -n verify-audit-logging -f -
apiVersion: v1
kind: Pod
metadata:
  name: ${name}
spec:
  containers:
    - name: ${name}
      image: registry.access.redhat.com/rhel7/rhel-tools
```

```
    command: ["/bin/sh", "-c"]
    args:
      ["sleep inf"]
EOF
done
```

成功输出会列出两个 pod，如 **pod/client** 和 **pod/server**，状态为 **created**。

4. 要生成流量并生成网络策略审计日志条目，请完成以下步骤：

   a. 在 **verify-audit-logging** 命名空间中获取名为 **server** 的 pod 的 IP 地址：

   ```
   $ POD_IP=$(oc get pods server -n verify-audit-logging -o jsonpath='{.status.podIP}')
   ```

   b. 从 **default** 命名空间中名为 **client** 的 pod 中 ping 上一个命令的 IP 地址，并确认所有数据包都已丢弃：

   ```
   $ oc exec -it client -n default -- /bin/ping -c 2 $POD_IP
   ```

   输出示例

   ```
   PING 10.128.2.55 (10.128.2.55) 56(84) bytes of data.

   --- 10.128.2.55 ping statistics ---
   2 packets transmitted, 0 received, 100% packet loss, time 2041ms
   ```

   c. 从 **verify-audit-logging** 命名空间中的客户端 pod 中，ping **POD_IP shell** 环境变量中存储的 IP 地址，并确认系统允许所有数据包。

   ```
   $ oc exec -it client -n verify-audit-logging -- /bin/ping -c 2 $POD_IP
   ```

   输出示例

   ```
   PING 10.128.0.86 (10.128.0.86) 56(84) bytes of data.
   64 bytes from 10.128.0.86: icmp_seq=1 ttl=64 time=2.21 ms
   64 bytes from 10.128.0.86: icmp_seq=2 ttl=64 time=0.440 ms

   --- 10.128.0.86 ping statistics ---
   2 packets transmitted, 2 received, 0% packet loss, time 1001ms
   rtt min/avg/max/mdev = 0.440/1.329/2.219/0.890 ms
   ```

5. 显示网络策略审计日志中的最新条目：

```
$ for pod in $(oc get pods -n openshift-ovn-kubernetes -l app=ovnkube-node --no-headers=true | awk '{ print $1 }') ; do
    oc exec -it $pod -n openshift-ovn-kubernetes -- tail -4 /var/log/ovn/acl-audit-log.log
  done
```

输出示例

```
2023-11-02T16:28:54.139Z|00004|acl_log(ovn_pinctrl0)|INFO|name="NP:verify-audit-logging:Ingress", verdict=drop, severity=alert, direction=to-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:81:02:01,dl_dst=0a:58:0a:81:02:23,nw_src=10.131.0.39,n
```

```
w_dst=10.129.2.35,nw_tos=0,nw_ecn=0,nw_ttl=62,nw_frag=no,tp_src=58496,tp_dst=8080,tcp
_flags=syn
2023-11-02T16:28:55.187Z|00005|acl_log(ovn_pinctrl0)|INFO|name="NP:verify-audit-
logging:Ingress", verdict=drop, severity=alert, direction=to-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:81:02:01,dl_dst=0a:58:0a:81:02:23,nw_src=10.131.0.39,n
w_dst=10.129.2.35,nw_tos=0,nw_ecn=0,nw_ttl=62,nw_frag=no,tp_src=58496,tp_dst=8080,tcp
_flags=syn
2023-11-02T16:28:57.235Z|00006|acl_log(ovn_pinctrl0)|INFO|name="NP:verify-audit-
logging:Ingress", verdict=drop, severity=alert, direction=to-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:81:02:01,dl_dst=0a:58:0a:81:02:23,nw_src=10.131.0.39,n
w_dst=10.129.2.35,nw_tos=0,nw_ecn=0,nw_ttl=62,nw_frag=no,tp_src=58496,tp_dst=8080,tcp
_flags=syn
2023-11-02T16:49:57.909Z|00028|acl_log(ovn_pinctrl0)|INFO|name="NP:verify-audit-
logging:allow-from-same-namespace:Egress:0", verdict=allow, severity=alert, direction=from-
lport:
icmp,vlan_tci=0x0000,dl_src=0a:58:0a:81:02:22,dl_dst=0a:58:0a:81:02:23,nw_src=10.129.2.34,
nw_dst=10.129.2.35,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,icmp_type=8,icmp_code=0
2023-11-02T16:49:57.909Z|00029|acl_log(ovn_pinctrl0)|INFO|name="NP:verify-audit-
logging:allow-from-same-namespace:Ingress:0", verdict=allow, severity=alert, direction=to-
lport:
icmp,vlan_tci=0x0000,dl_src=0a:58:0a:81:02:22,dl_dst=0a:58:0a:81:02:23,nw_src=10.129.2.34,
nw_dst=10.129.2.35,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,icmp_type=8,icmp_code=0
2023-11-02T16:49:58.932Z|00030|acl_log(ovn_pinctrl0)|INFO|name="NP:verify-audit-
logging:allow-from-same-namespace:Egress:0", verdict=allow, severity=alert, direction=from-
lport:
icmp,vlan_tci=0x0000,dl_src=0a:58:0a:81:02:22,dl_dst=0a:58:0a:81:02:23,nw_src=10.129.2.34,
nw_dst=10.129.2.35,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,icmp_type=8,icmp_code=0
2023-11-02T16:49:58.932Z|00031|acl_log(ovn_pinctrl0)|INFO|name="NP:verify-audit-
logging:allow-from-same-namespace:Ingress:0", verdict=allow, severity=alert, direction=to-
lport:
icmp,vlan_tci=0x0000,dl_src=0a:58:0a:81:02:22,dl_dst=0a:58:0a:81:02:23,nw_src=10.129.2.34,
nw_dst=10.129.2.35,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,icmp_type=8,icmp_code=0
```

4.6. 为命名空间启用出口防火墙和网络策略审计日志

作为集群管理员，您可以为命名空间启用审计日志。

先决条件

- 安装 OpenShift CLI（**oc**）。
- 使用具有 **cluster-admin** 权限的用户登陆到集群。

流程

- 要为命名空间启用审计日志，请输入以下命令：

```
$ oc annotate namespace <namespace> \
  k8s.ovn.org/acl-logging='{ "deny": "alert", "allow": "notice" }'
```

其中：

<namespace>
指定命名空间的名称。

提示

您还可以应用以下 YAML 来启用审计日志记录：

```
kind: Namespace
apiVersion: v1
metadata:
  name: <namespace>
  annotations:
    k8s.ovn.org/acl-logging: |-
      {
        "deny": "alert",
        "allow": "notice"
      }
```

成功输出列出了审计日志记录名称，状态为 **annotated**。

验证

- 显示审计日志中的最新条目：

```
$ for pod in $(oc get pods -n openshift-ovn-kubernetes -l app=ovnkube-node --no-headers=true | awk '{ print $1 }') ; do
    oc exec -it $pod -n openshift-ovn-kubernetes -- tail -4 /var/log/ovn/acl-audit-log.log
  done
```

输出示例

```
2023-11-02T16:49:57.909Z|00028|acl_log(ovn_pinctrl0)|INFO|name="NP:verify-audit-logging:allow-from-same-namespace:Egress:0", verdict=allow, severity=alert, direction=from-lport: icmp,vlan_tci=0x0000,dl_src=0a:58:0a:81:02:22,dl_dst=0a:58:0a:81:02:23,nw_src=10.129.2.34,nw_dst=10.129.2.35,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,icmp_type=8,icmp_code=0
2023-11-02T16:49:57.909Z|00029|acl_log(ovn_pinctrl0)|INFO|name="NP:verify-audit-logging:allow-from-same-namespace:Ingress:0", verdict=allow, severity=alert, direction=to-lport: icmp,vlan_tci=0x0000,dl_src=0a:58:0a:81:02:22,dl_dst=0a:58:0a:81:02:23,nw_src=10.129.2.34,nw_dst=10.129.2.35,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,icmp_type=8,icmp_code=0
2023-11-02T16:49:58.932Z|00030|acl_log(ovn_pinctrl0)|INFO|name="NP:verify-audit-logging:allow-from-same-namespace:Egress:0", verdict=allow, severity=alert, direction=from-lport: icmp,vlan_tci=0x0000,dl_src=0a:58:0a:81:02:22,dl_dst=0a:58:0a:81:02:23,nw_src=10.129.2.34,nw_dst=10.129.2.35,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,icmp_type=8,icmp_code=0
2023-11-02T16:49:58.932Z|00031|acl_log(ovn_pinctrl0)|INFO|name="NP:verify-audit-logging:allow-from-same-namespace:Ingress:0", verdict=allow, severity=alert, direction=to-lport: icmp,vlan_tci=0x0000,dl_src=0a:58:0a:81:02:22,dl_dst=0a:58:0a:81:02:23,nw_src=10.129.2.34,nw_dst=10.129.2.35,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,icmp_type=8,icmp_code=0
```

4.7. 为命名空间禁用出口防火墙和网络策略审计日志

作为集群管理员，您可以禁用命名空间的审计日志。

先决条件

先决条件

- 安装 OpenShift CLI（**oc**）。
- 使用具有 **cluster-admin** 权限的用户登陆到集群。

流程

- 要为命名空间禁用审计日志，请输入以下命令：

```
$ oc annotate --overwrite namespace <namespace> k8s.ovn.org/acl-logging-
```

其中：

<namespace>
指定命名空间的名称。

提示

您还可以应用以下 YAML 来禁用审计日志记录：

```
kind: Namespace
apiVersion: v1
metadata:
  name: <namespace>
  annotations:
    k8s.ovn.org/acl-logging: null
```

成功输出列出了审计日志记录名称，状态为 **annotated**。

4.8. 其他资源

- 关于网络策略
- 为项目配置出口防火墙

第 5 章 EGRESS 防火墙

5.1. 查看项目的出口防火墙

作为集群管理员，您可以列出任何现有出口防火墙的名称，并查看特定出口防火墙的流量规则。

5.1.1. 查看 EgressFirewall 自定义资源 (CR)

您可以查看集群中的 **EgressFirewall** CR。

先决条件

- 使用 OVN-Kubernetes 网络插件的集群。
- 安装 OpenShift 命令行界面 (CLI)，通常称为 **oc**。
- 您必须登录集群。

流程

1. 可选： 要查看集群中定义的 **EgressFirewall** CR 的名称，请输入以下命令：

```
$ oc get egressfirewall --all-namespaces
```

2. 要检查策略，请输入以下命令。将 **<policy_name>** 替换为要检查的策略名称。

```
$ oc describe egressfirewall <policy_name>
```

输出示例

```
Name:  default
Namespace: project1
Created: 20 minutes ago
Labels:  <none>
Annotations: <none>
Rule:  Allow to 1.2.3.0/24
Rule:  Allow to www.example.com
Rule:  Deny to 0.0.0.0/0
```

5.2. 为项目编辑出口防火墙

作为集群管理员，您可以修改现有出口防火墙的网络流量规则。

5.2.1. 编辑 EgressFirewall 自定义资源 (CR)

作为集群管理员，您可以更新一个项目的出口防火墙。

先决条件

- 使用 OVN-Kubernetes 网络插件的集群。

- 安装 OpenShift CLI (**oc**)。
- 您需要使用集群管理员身份登陆到集群。

流程

1. 查找项目的 **EgressFirewall** CR 的名称。将 **<project>** 替换为项目的名称。

```
$ oc get -n <project> egressfirewall
```

2. 可选：如果您在创建出口网络防火墙时没有保存 **EgressFirewall** 对象的副本，请输入以下命令来创建副本。

```
$ oc get -n <project> egressfirewall <name> -o yaml > <filename>.yaml
```

将 **<project>** 替换为项目的名称。将 **<name>** 替换为 Pod 的名称。将 **<filename>** 替换为要将 YAML 保存到的文件的名称。

3. 更改策略规则后，输入以下命令替换 **EgressFirewall** CR。将 **<filename>** 替换为包含更新的 **EgressFirewall** CR 的文件名称。

```
$ oc replace -f <filename>.yaml
```

5.3. 从项目中删除出口防火墙

作为集群管理员，您可以从项目中删除出口防火墙，从而删除对项目的离开 OpenShift Container Platform 集群的网络流量的限制。

5.3.1. 删除 EgressFirewall CR

作为集群管理员，您可以从项目中删除出口防火墙。

先决条件

- 使用 OVN-Kubernetes 网络插件的集群。
- 安装 OpenShift CLI (**oc**)。
- 您需要使用集群管理员身份登陆到集群。

流程

1. 查找项目的 **EgressFirewall** CR 的名称。将 **<project>** 替换为项目的名称。

```
$ oc get egressfirewall -n <project>
```

2. 输入以下命令删除 **EgressFirewall** CR。将 **<project>** 替换为项目名称，**<name>** 替换为对象名称。

```
$ oc delete -n <project> egressfirewall <name>
```

5.4. 为项目配置出口防火墙

作为集群管理员，您可以为项目创建一个出口防火墙，用于限制离开 OpenShift Container Platform 集群的出口流量。

5.4.1. 出口防火墙在一个项目中的工作原理

作为集群管理员，您可以使用一个*出口防火墙*来限制集群内的一些 pod 或所有 pod 可以访问的外部主机。出口防火墙适用于以下情况：

- pod 只能连接到内部主机，且无法启动到公共互联网的连接。
- pod 只能连接到公共互联网，且无法启动到 OpenShift Container Platform 集群以外的内部主机的连接。
- pod 无法访问 OpenShift Container Platform 集群外的特定内部子网或主机。
- pod 只能连接到特定的外部主机。

例如，您可以允许某一个项目访问指定的 IP 范围，但拒绝其他项目对同一 IP 范围的访问。或者您可以限制应用程序开发人员从 Python pip 的镜像点进行更新，并强制要求更新只能来自于批准的源。

您可以通过创建一个 **EgressFirewall** 自定义资源 (CR) 来配置出口防火墙策略。出口防火墙与满足以下任一条件的网络流量匹配：

- CIDR 格式的 IP 地址范围
- 解析为 IP 地址的 DNS 名称
- 端口号
- 协议是以下协议之一： TCP、UDP 和 SCTP

5.4.1.1. 出口防火墙的限制

出口防火墙有以下限制：

- 项目不能有多个 **EgressFirewall** CR。
- 出口防火墙规则不适用于通过路由器的网络流量。任何有权创建 **Route** CR 对象的用户，都可以通过创建指向禁止的目的地的路由来绕过出口防火墙策略规则。
- 出口防火墙不适用于主机网络命名空间。启用主机网络的 Pod 不受出口防火墙规则的影响。
- 如果您的出口防火墙包含 **0.0.0.0/0** 的拒绝规则，则阻止访问 OpenShift Container Platform API 服务器。您必须为每个 IP 地址添加允许规则，或使用出口策略规则中的 **nodeSelector** 类型允许规则来连接到 API 服务器。
以下示例演示了确保 API 服务器访问所需的出口防火墙规则的顺序：

```
apiVersion: k8s.ovn.org/v1
kind: EgressFirewall
metadata:
  name: default
  namespace: <namespace>
spec:
  egress:
```

```
  - to:
      cidrSelector: <api_server_address_range> (1)
    type: Allow
# ...
  - to:
      cidrSelector: 0.0.0.0/0 (2)
    type: Deny
```

其中：

<namespace>
: 指定出口防火墙的命名空间。

<api_server_address_range>
: 指定包含 OpenShift Container Platform API 服务器的 IP 地址范围。

<cidrSelector>
: 指定 **0.0.0.0/0** 值来设置全局拒绝规则，阻止访问 OpenShift Container Platform API 服务器。
要查找 API 服务器的 IP 地址，请运行 **oc get ep kubernetes -n default**。

如需更多信息，请参阅 BZ#1988324。

- 每个项目最多可定义一个具有最多 8,000 个规则的 **EgressFirewall** 对象。
- 如果您在 Red Hat OpenShift Networking 中使用带有共享网关模式的 OVN-Kubernetes 网络插件，则返回入口回复会受到出口防火墙规则的影响。如果出口防火墙规则丢弃入口回复目的地 IP，流量将被丢弃。
- 通常，在出口防火墙策略中使用域名服务器(DNS)名称不会影响通过 CoreDNS 进行本地 DNS 解析。但是，如果您的出口防火墙策略使用域名，并且外部 DNS 服务器处理受影响 pod 的 DNS 解析，则必须包括允许访问 DNS 服务器的 IP 地址的出口防火墙规则。

违反这些限制会导致项目的出口防火墙出现问题。因此，所有外部网络流量都会被丢弃，这可能会给您的组织造成安全风险。

kube-node-lease、**kube-public**、**kube-system**、**openshift** 和 **openshift-** 项目中创建一个 **EgressFirewall** 资源。

5.4.1.2. 出口防火墙策略规则的匹配顺序

OVN-Kubernetes 根据定义的顺序评估出口防火墙策略规则，从第一个到最后一个的顺序。第一个与 pod 的出口连接匹配的规则会被应用。该连接会忽略后续的所有规则。

5.4.1.3. 域名服务器 (DNS) 解析如何工作

如果您在 egress 防火墙策略规则中使用 DNS 名称，则正确解析域名会受到以下限制：

- 域名更新会根据生存时间（TTL）持续时间进行轮询。默认情况下，持续时间为 30 分钟。当出口防火墙控制器查询本地名称服务器以获取域名时，如果响应包含 TTL 且 TTL 小于 30 分钟，控制器会将该 DNS 名称的持续时间设置为返回的值。每个 DNS 名称都会在 DNS 记录的 TTL 过期后查询。

- 在需要时，pod 必须通过相同的本地名称服务器解析域名。否则，egress 防火墙控制器和 pod 已知的域的 IP 地址可能会有所不同。如果主机名的 IP 地址不同，则出口防火墙的强制实施可能不一致。
- 因为出口防火墙控制器和 pod 异步轮询相同的本地名称服务器，所以 pod 可能会在出口控制器执行前获取更新的 IP 地址，从而导致竞争条件。由于这个限制，仅建议在 **EgressFirewall** 对象中使用域名来更改 IP 地址的域。

5.4.1.3.1. 改进了 DNS 解析并解析通配符域名

在某些情况下，与 DNS 记录关联的 IP 地址会频繁更改，或者您可能希望在出口（egress）防火墙策略规则中指定通配符域名。

在这种情况下，OVN-Kubernetes 集群管理器为每个出口防火墙策略规则中使用的每个唯一 DNS 名称创建一个 **DNSNameResolver** 自定义资源对象。此自定义资源存储以下信息：

重要

改进了出口防火墙规则的 DNS 解析只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能支持范围的更多信息，请参阅以下链接：

- 技术预览功能支持范围

DNSNameResolver CR 定义示例

```
apiVersion: networking.openshift.io/v1alpha1
kind: DNSNameResolver
spec:
  name: www.example.com.
status:
  resolvedNames:
  - dnsName: www.example.com.
    resolvedAddress:
    - ip: "1.2.3.4"
      ttlSeconds: 60
      lastLookupTime: "2023-08-08T15:07:04Z"
```

其中：

<name>
: 指定 DNS 名称。这可以是标准 DNS 名称或通配符 DNS 名称。对于通配符 DNS 名称，DNS 名称解析信息包含与通配符 DNS 名称匹配的所有 DNS 名称。

<dnsName>
: 指定与 **spec.name** 字段匹配的解析 DNS 名称。如果 **spec.name** 字段包含通配符 DNS 名称，则会创建多个 **dnsName** 条目，其中包含在解析时与通配符 DNS 名称匹配的标准 DNS 名称。如果通配符 DNS 名称也可以成功解析，则此字段也存储通配符 DNS 名称。<ip> 指定与 DNS 名称关联的当前 IP 地址。

<ttlSeconds>
: 指定最后一次生存时间(TTL)持续时间。

<lastLookupTime>
: 指定最后一次查找时间。

如果在 DNS 解析 DNS 名称的过程中，查询中的 DNS 名称与 **DNSNameResolver** CR 中定义的任何名称匹配，那么在 CR **status** 字段中会相应地更新之前的信息。如果 DNS 通配符名称查找失败，会在默认的 TTL 为 30 分钟后重试请求。

OVN-Kubernetes 集群管理器监视对 **EgressFirewall** 自定义资源对象的更新，并在更新发生时创建、修改或删除与这些出口防火墙策略关联的 **DNSNameResolver** CR。

> **警告**
>
> 不要直接修改 **DNSNameResolver** 自定义资源。这可能导致出口防火墙的不需要的行为。

5.4.2. EgressFirewall 自定义资源 (CR)

您可以为出口防火墙定义一个或多个规则。规则是一个 **Allow** 规则，也可以是一个 **Deny** 规则，它包括规则适用的流量规格。

以下 YAML 描述了 **EgressFirewall** CR：

EgressFirewall 对象

```
apiVersion: k8s.ovn.org/v1
kind: EgressFirewall
metadata:
  name: <ovn>
spec:
  egress: <egress_rules>
    ...
```

其中：

<ovn>
: 对象的名称必须是 **default**。

<egress_rules>
: 如下小节所述，指定一个或多个出口网络策略规则的集合。

5.4.2.1. EgressFirewall 规则

以下 YAML 描述了 **EgressFirewall** 资源的规则。用户可以选择 CIDR 格式的 IP 地址范围、域名，或使用 **nodeSelector** 字段来允许或拒绝出口流量。**egress** 小节需要一个包括一个或多个对象的数组。

出口策略规则小节

```
egress:
- type: <type>
```

```
    to:
      cidrSelector: <cidr_range>
      dnsName: <dns_name>
      nodeSelector: <label_name>: <label_value>
    ports: <optional_port>
      ...
```

其中：

<type>
指定规则的类型。该值必须是 **Allow** 或 **Deny**。

<to>
指定描述出口流量匹配规则的小节，该规则指定 **cidrSelector** 字段或 **dnsName** 字段。您不能在同一规则中使用这两个字段。

<cidr_range>
指定 CIDR 格式的 IP 地址范围。

<dns_name>
指定 DNS 域名。

<nodeSelector>
指定标签，它们是用户定义的键和值对。标签附加到对象，如 pod。**nodeSelector** 允许选择一个或多个节点标签，并附加到 pod。

<ports>
指定描述规则的网络端口和协议集合的可选字段。

端口小节

```
ports:
- port:
  protocol:
```

其中：

<port>
指定网络端口，如 **80** 或 **443**。如果为此字段指定一个值，还必须为 **protocol** 字段指定一个值。

<protocol>
指定网络协议。该值必须是 **TCP**、**UDP** 或 **SCTP**。

5.4.2.2. EgressFirewall CR 示例

以下示例定义了几个出口防火墙策略规则：

```
apiVersion: k8s.ovn.org/v1
kind: EgressFirewall
metadata:
  name: default
spec:
  egress: (1)
  - type: Allow
    to:
      cidrSelector: 1.2.3.0/24
```

```
  - type: Deny
    to:
      cidrSelector: 0.0.0.0/0
```

其中：

<egress>
指定出口防火墙策略规则对象的集合。

以下示例定义了一个策略规则，即如果流量使用 TCP 协议和目标端口 **80**，或任何协议和目标端口 **443**，则拒绝通过 **172.16.1.1/32** IP 地址到主机的流量。

```
apiVersion: k8s.ovn.org/v1
kind: EgressFirewall
metadata:
  name: default
spec:
  egress:
  - type: Deny
    to:
      cidrSelector: 172.16.1.1/32
    ports:
    - port: 80
      protocol: TCP
    - port: 443
```

5.4.2.3. 使用 nodeSelector 的 EgressFirewall CR 示例

作为一个集群管理员，您可以使用 **nodeSelector** 字段指定标签来允许或拒绝集群中节点的出口流量。标签可以应用到一个或多个节点。标签很有用，因为不是为每个节点 IP 地址添加手动规则，所以您可以使用节点选择器创建一个标签，以允许出口防火墙后面的 pod 访问主机网络 pod。以下是带有 **region=east** 标签的示例：

```
apiVersion: k8s.ovn.org/v1
kind: EgressFirewall
metadata:
  name: default
spec:
    egress:
    - to:
        nodeSelector:
          matchLabels:
            region: east
      type: Allow
```

5.4.3. 创建 EgressFirewall 自定义资源 (CR)

作为集群管理员，您可以为项目创建一个出口防火墙策略对象。

重要

如果项目已有 **EgressFirewall** 资源，您必须编辑现有策略来更改出口防火墙规则。

先决条件

- 使用 OVN-Kubernetes 网络插件的集群。
- 安装 OpenShift CLI (**oc**) 。
- 您需要使用集群管理员身份登陆到集群。

流程

1. 创建策略规则：
   a. 创建一个 **<policy_name>.yaml** 文件，其中 **<policy_name>** 描述出口策略规则。
   b. 在文件中定义 **EgressFirewall** 对象。
2. 运行以下命令来创建策略对象。将 **<policy_name>** 替换为策略的名称， **<project>** 替换为规则应用到的项目。

```
$ oc create -f <policy_name>.yaml -n <project>
```

   成功输出列出了 **egressfirewall.k8s.ovn.org/v1** 名称以及 **created** 状态。
3. 可选：保存 **<policy_name>.yaml** 文件，以便在以后进行修改。

第 6 章 配置 IPSEC 加密

通过启用 IPsec，您可以加密节点之间的内部 pod 到 pod 集群流量，以及集群外部的 pod 和 IPsec 端点之间的外部流量。OVN-Kubernetes 集群网络上的节点之间的所有 pod 到 pod 网络流量都使用 IPsec *传输模式* 加密。

默认禁用 IPsec。您可以在安装集群安装过程中或安装后启用 IPsec。有关集群安装的详情，请参阅 OpenShift Container Platform 安装概述。

注意

当 **libreswan** 和 **NetworkManager-libreswan** 软件包有不同的 OpenShift Container Platform 版本时，将集群升级到 OpenShift Container Platform 4.19 会导致两个连续的计算节点重新引导操作。对于第一次重启，Cluster Network Operator (CNO)将 IPsec 配置应用到计算节点。对于第二个重启，Machine Config Operator (MCO)会将最新的机器配置应用到集群。

要将 CNO 和 MCO 更新合并到单一节点重启后，请完成以下任务：

- 在升级集群前，在对计算节点进行分组的 **MachineConfigPools** 自定义资源(CR)中将 **paused** 参数设置为 **true**。
- 在升级集群后，将参数设置为 **false**。

如需更多信息，请参阅 执行 Control Plane Only 更新 。

OpenShift Container Platform 集群中 IPsec 存在以下支持限制：

- 在 IBM Cloud® 上，IPsec 仅支持网络地址转换(NAT-T)。在此平台上不支持 Encapsulating Security Payload (ESP)。
- 如果您的集群在 Red Hat OpenShift Container Platform 中使用 托管的 control plane，则 IPsec 不支持 pod 到 pod，以及到外部主机的 IPsec 加密。
- 如果有一个或多个接口附加到 Open vSwitch (OVS)，则不支持在任何网络接口中使用 ESP 硬件卸载。为集群启用 IPsec 会触发将 IPsec 与附加到 OVS 的接口搭配使用。默认情况下，OpenShift Container Platform 在附加到 OVS 的任何接口上禁用 ESP 硬件卸载。
- 如果您为未附加到 OVS 的网络接口启用了 IPsec，集群管理员必须在未附加到 OVS 的每个接口上手动禁用 ESP 硬件卸载。

以下列表概述了 IPsec 文档中的关键任务：

- 在集群安装后启用和禁用 IPsec。
- 为集群和外部主机之间的流量配置 IPsec 加密。
- 验证 IPsec 是否加密不同节点上的 pod 之间的流量。

6.1. 操作模式

在 OpenShift Container Platform 集群中使用 IPsec 时，您可以从以下操作模式中选择：

表 6.1. 操作的 IPsec 模式

模式	描述	default
Disabled	没有流量被加密。这是集群的默认设置。	是
Full	pod 到 pod 的流量被加密，如"由 pod 到 pod IPsec 加密的网络流量类型"中所述。完成 IPsec 所需的配置步骤后，可以加密到外部节点的流量。	否
外部	完成 IPsec 所需的配置步骤后，可以加密到外部节点的流量。	否

6.2. 先决条件

对于将流量加密到外部主机的 IPsec 支持，请确保满足以下先决条件：

- 在 OVN-Kubernetes 网络插件的 **ovnKubernetesConfig.gatewayConfig** 规格中设置 **routingViaHost=true**。
- 安装 NMState Operator。指定 IPsec 配置需要这个 Operator。如需更多信息，请参阅 Kubernetes NMState Operator。

  注意

  Google Cloud 仅支持 NMState Operator 来配置 IPsec。

- 已安装 Butane 工具 (**butane**)。要安装 Butane，请参阅安装 Butane。

这些先决条件需要将证书添加到主机 NSS 数据库中，并配置 IPsec 与外部主机进行通信。

6.3. 启用 IPSEC 时的网络连接要求

您必须配置机器之间的网络连接，以允许 OpenShift Container Platform 集群组件进行通信。每台机器都必须能够解析集群中所有其他机器的主机名。

表 6.2. 用于全机器到所有机器通信的端口

协议	port	描述
UDP	**500**	IPsec IKE 数据包
	4500	IPsec NAT-T 数据包
ESP	N/A	IPsec Encapsulating Security Payload(ESP)

6.4. POD 到 POD 流量的 IPSEC 加密

对于 pod 到 pod 流量的 IPsec 加密，以下小节描述了加密哪些特定的 pod 到 pod 的流量，使用什么加密协议，以及如何处理 X.509 证书。这些部分不适用于集群和外部主机之间的 IPsec 加密，您必须为特定的外部网络基础架构手动配置。

6.4.1. 由 pod 到 pod IPsec 加密的网络流量类型

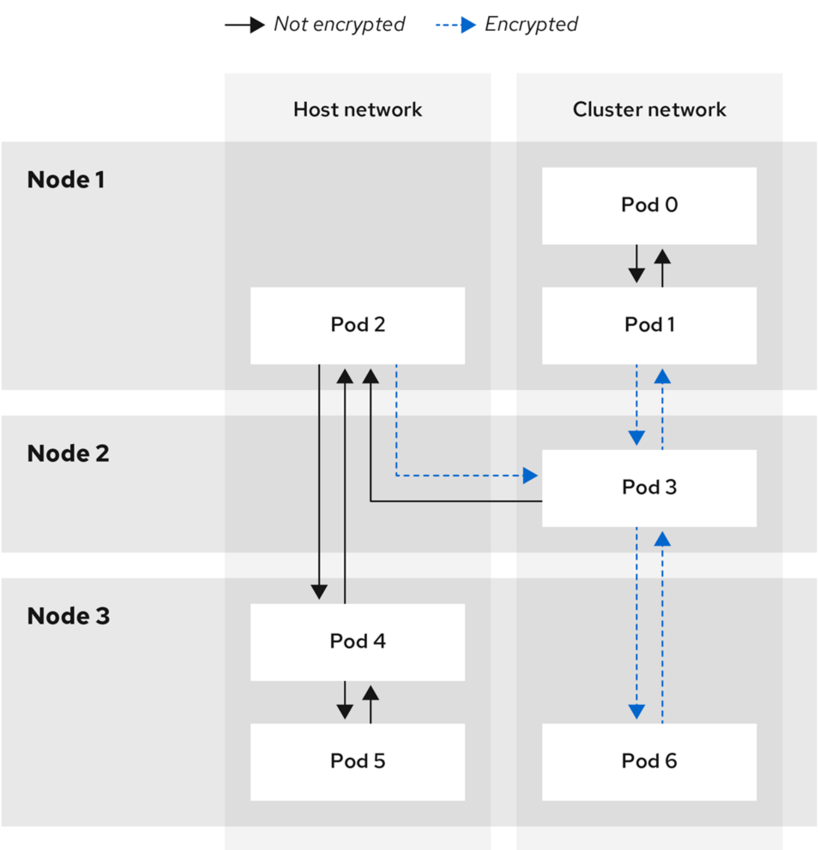
启用 IPsec 后，只有 pod 间的以下网络流量会被加密：

- 集群网络的不同节点上的 pod 间的流量
- 从主机网络上的 pod 流量到集群网络上的 pod

以下流量流没有加密：

- 集群网络上同一节点上的 pod 间的流量
- 主机网络上的 pod 间的流量
- 从集群网络上的 pod 流量到主机网络上的 pod

下图中显示了加密和未加密的流程：



6.4.2. 加密协议和 IPsec 模式

使用的加密机制是 **AES-GCM-16-256**。完整性检查值（ICV）为 **16** 字节。密钥长度为 **256** 位。

使用的 IPsec 模式是 *传输模式*，这是通过向原始数据包的 IP 标头添加封装安全 Payload (ESP) 标头来加密端到端通信的模式。OpenShift Container Platform 目前不支持 pod 到 pod 通信的 IPsec *Tunnel 模式*。

6.4.3. 安全证书生成和轮转

Cluster Network Operator（CNO）生成自签名 X.509 证书颁发机构（CA），该颁发机构（CA）用于加密。来自每个节点的证书签名请求（CSR）由 CNO 自动实现。

CA 的有效期为 10 年。独立节点证书的有效期为 5 年，并在 4 年半后自动轮转。

6.5. 外部流量的 IPSEC 加密

OpenShift Container Platform 支持使用 IPsec 来加密用于外部主机的流量，确保传输过程中数据的保密性和完整性。此功能依赖于您必须提供的 X.509 证书。

6.5.1. 支持的平台

在以下平台上支持此功能：

- 裸机
- Google Cloud
- Red Hat OpenStack Platform (RHOSP)
- VMware vSphere

重要

如果您有 Red Hat Enterprise Linux (RHEL) 计算（compute）节点，它们不支持外部流量的 IPsec 加密。

如果您的集群在 Red Hat OpenShift Container Platform 中使用托管的 control plane，则不支持将流量加密到外部主机的 IPsec。

6.5.2. 限制：

确保观察到以下限制：

- 在为外部流量配置 IPsec 时，NMState Operator 目前不支持 IPv6 配置。
- 提供的证书捆绑包中的证书通用名称(CN)不能以 **ovs_** 前缀开头，因为这个命名可以与每个节点的网络安全服务(NSS)数据库中的 pod 到 pod 的 IPsec CN 名称冲突。

6.6. 启用 IPSEC 加密

作为集群管理员，您可以在集群和外部 IPsec 端点之间启用 pod 到 pod 的 IPsec 加密。

您可以使用以下模式之一配置 IPsec：

- **Full** ：pod 到 pod 和外部流量的加密
- **External** ：对外部流量进行加密

注意

如果以 **Full** 模式配置 IPsec，还必须完成"为外部流量配置 IPsec 加密"过程。

如果以 **Full** 模式启用 IPsec，作为集群管理员，您可以通过将 **full** 模式添加到 **networks.operator.openshift.io** 来配置模式的选项。**full** 模式支持 **encapsulation** 参数。您可以使用此参数为 IPsec 流量配置网络地址转换(NAT-T) 封装。**encapsulation** 参数支持以下值：

- **Auto** 是默认值，当 **libreswan** 检测到节点内流量中的网络地址转换(NAT)数据包时启用 UDP 封装。
- **Always** 为节点中所有流量类型启用 UDP 封装。这个选项不依赖于 **libreswan** 来检测节点中的 NAT 数据包。

先决条件

- 安装 OpenShift CLI (**oc**) 。
- 以具有 **cluster-admin** 权限的用户身份登录集群。
- 您已将集群 MTU 的大小减少为 **46** 字节，以允许 IPsec ESP 标头的开销。

流程

1. 要启用 IPsec 加密，请输入以下命令：

```
$ oc patch networks.operator.openshift.io cluster --type=merge -p \
 '{
 "spec":{
   "defaultNetwork":{
     "ovnKubernetesConfig":{
       "ipsecConfig":{
         "mode":"<mode"> 1
       }}}}}'
```

1 1 指定 **External** 以加密到外部主机的流量，或者指定 **Full** 来加密 pod 到 pod 流量，以及可选的到外部主机的流量。默认情况下禁用 IPsec。

以 Full 模式启用 IPsec 的配置示例，并将 encapsulation 设置为 Always

```
$ oc patch networks.operator.openshift.io cluster --type=merge -p \
 '{
 "spec":{
   "defaultNetwork":{
     "ovnKubernetesConfig":{
       "ipsecConfig":{
         "mode":"Full",
         "full":{
           "encapsulation": "Always"
         }}}}}}'
```

2. 完成"为外部流量配置 IPsec 加密"流程，使用 IPsec 加密外部流量。

验证

1. 要查找 OVN-Kubernetes data plane pod 的名称，请输入以下命令：

```
$ oc get pods -n openshift-ovn-kubernetes -l=app=ovnkube-node
```

输出示例

```
ovnkube-node-5xqbf                        8/8     Running   0           28m
ovnkube-node-6mwcx                        8/8     Running   0           29m
ovnkube-node-ck5fr                        8/8     Running   0           31m
ovnkube-node-fr4ld                        8/8     Running   0           26m
ovnkube-node-wgs4l                        8/8     Running   0           33m
ovnkube-node-zfvcl                        8/8     Running   0           34m
...
```

2. 运行以下命令，验证集群中是否启用了 IPsec：

注意

作为集群管理员，您可以在 IPsec 配置为 **Full** 模式时验证集群中 pod 之间是否启用了 IPsec。此步骤不会验证 IPsec 是否在集群和外部主机间工作。

```
$ oc -n openshift-ovn-kubernetes rsh ovnkube-node-<XXXXX> ovn-nbctl --no-leader-only get nb_global . ipsec (1)
```

其中：**<XXXXX>** 指定上一步中 pod 的随机字符序列。

命令的成功输出显示状态为 **true**。

6.7. 为外部流量配置 IPSEC 加密

作为集群管理员，要使用 IPsec 加密外部流量，您必须为网络基础架构配置 IPsec，包括提供 PKCS#12 证书。因为这个流程使用 Butane 创建机器配置，所以必须安装 **butane** 工具。

注意

应用机器配置后，Machine Config Operator（MCO）会重新引导集群中受影响的节点，以推出新机器配置。

先决条件

- 安装 OpenShift CLI (**oc**) 。
- 您已在本地计算机上安装了 **butane** 工具。
- 在集群中安装了 NMState Operator。
- 以具有 **cluster-admin** 权限的用户身份登录集群。
- 您有一个 IPsec 端点的现存的 PKCS#12 证书，以及一个 Privacy Enhanced Mail (PEM) 格式的 CA 证书。
- 您在集群中的 **Full** 或 **External** 模式中启用了 IPsec。
- 您必须在 OVN-Kubernetes 网络插件的 **ovnKubernetesConfig.gatewayConfig** 规格中将 **routingViaHost** 参数设置为 **true**。

流程

1. 使用 NMState Operator 节点网络配置策略创建 IPsec 配置。如需更多信息，请参阅使用 nmstatectl 配置基于 IPsec 的 VPN 连接。

   a. 要识别 IPsec 端点的集群节点的 IP 地址，请输入以下命令：

   ```
   $ oc get nodes
   ```

   b. 创建名为 **ipsec-config.yaml** 的文件，其中包含 NMState Operator 的节点网络配置策略，如下例所示。有关 **NodeNetworkConfigurationPolicy** 对象的概述，请参阅 Kubernetes NMState 项目。

NMState IPsec 传输配置示例

```
apiVersion: nmstate.io/v1
kind: NodeNetworkConfigurationPolicy
metadata:
  name: ipsec-config
spec:
  nodeSelector:
    kubernetes.io/hostname: "<hostname>"
  desiredState:
    interfaces:
    - name: <interface_name>
      type: ipsec
      libreswan:
        left: <cluster_node>
        leftid: '%fromcert'
        leftrsasigkey: '%cert'
        leftcert: left_server
        leftmodecfgclient: false
        right: <external_host>
        rightid: '%fromcert'
        rightrsasigkey: '%cert'
        rightsubnet: <external_address>/32
        ikev2: insist
        type: transport
```

其中：

kubernetes.io/hostname
指定要将策略应用到的主机名。此主机充当 IPsec 配置中的左侧主机。

name
指定要在主机上创建的接口名称。

left
指定在集群端终止 IPsec 隧道的集群节点的主机名。名称应与您提供的 PKCS#12 证书中的 SAN **[Subject Alternate Name]** 匹配。

right
指定外部主机名，如 **host.example.com**。名称应与您提供的 PKCS#12 证书中的 SAN **[Subject Alternate Name]** 匹配。

rightsubnet
指定外部主机的 IP 地址，如 **10.1.2.3/32**。

NMState IPsec 隧道配置示例

```yaml
apiVersion: nmstate.io/v1
kind: NodeNetworkConfigurationPolicy
metadata:
  name: ipsec-config
spec:
  nodeSelector:
    kubernetes.io/hostname: "<hostname>"
  desiredState:
    interfaces:
    - name: <interface_name>
      type: ipsec
      libreswan:
        left: <cluster_node>
        leftid: '%fromcert'
        leftmodecfgclient: false
        leftrsasigkey: '%cert'
        leftcert: left_server
        right: <external_host>
        rightid: '%fromcert'
        rightrsasigkey: '%cert'
        rightsubnet: <external_address>/32
        ikev2: insist
        type: tunnel
```

c. 要配置 IPsec 接口，请输入以下命令：

```
$ oc create -f ipsec-config.yaml
```

2. 提供以下证书文件以添加到每个主机上的网络安全服务(NSS)数据库中。这些文件在后续步骤中作为 Butane 配置的一部分导入。
   - **left_server.p12**：IPsec 端点的证书捆绑包
   - **ca.pem** ：您使用签名证书的证书颁发机构
3. 创建机器配置以将证书添加到集群中。
4. 从挂载的 secret 文件中读取密码：

```
$ password=$(cat run/secrets/<left_server_password>)
```

   - **left_server_password**:: 包含密码的文件名称。此文件存在于挂载的 secret 中。
5. 使用 **pk12util** 工具，它预先打包了 Red Hat Enterprise Linux (RHEL)，输入以下命令指定保护 **PKCS#12** 文件的密码。确保将 **<password>** 值替换为您的密码。

```
$ pk12util -W "<password>" -i /etc/pki/certs/left_server.p12 -d /var/lib/ipsec/nss/
```

6. 要为 control plane 和计算节点创建 Butane 配置文件，请输入以下命令：

注意

您在配置文件中指定的 Butane 版本应与 OpenShift Container Platform 版本匹配，并且始终以 0 结尾。例如，**4.19.0**。有关 Butane 的信息，请参阅"使用 Butane 创建机器配置"。

```
$ for role in master worker; do
  cat >> "99-ipsec-${role}-endpoint-config.bu" <<-EOF
  variant: openshift
  version: 4.19.0
  metadata:
    name: 99-${role}-import-certs
    labels:
      machineconfiguration.openshift.io/role: $role
  systemd:
    units:
    - name: ipsec-import.service
      enabled: true
      contents: |
        [Unit]
        Description=Import external certs into ipsec NSS
        Before=ipsec.service

        [Service]
        Type=oneshot
        ExecStart=/usr/local/bin/ipsec-addcert.sh
        RemainAfterExit=false
        StandardOutput=journal

        [Install]
        WantedBy=multi-user.target
  storage:
    files:
    - path: /etc/pki/certs/ca.pem
      mode: 0400
      overwrite: true
      contents:
        local: ca.pem
    - path: /etc/pki/certs/left_server.p12
      mode: 0400
      overwrite: true
      contents:
        local: left_server.p12
    - path: /usr/local/bin/ipsec-addcert.sh
      mode: 0740
      overwrite: true
      contents:
        inline: |
          #!/bin/bash -e
          echo "importing cert to NSS"
          certutil -A -n "CA" -t "CT,C,C" -d /var/lib/ipsec/nss/ -i /etc/pki/certs/ca.pem
          pk12util -W "" -i /etc/pki/certs/left_server.p12 -d /var/lib/ipsec/nss/
          certutil -M -n "left_server" -t "u,u,u" -d /var/lib/ipsec/nss/
EOF
done
```

7. 要将上一步中创建的 Butane 文件转换为机器配置，请输入以下命令：

```
$ for role in master worker; do
  butane -d . 99-ipsec-${role}-endpoint-config.bu -o ./99-ipsec-$role-endpoint-config.yaml
done
```

8. 要将机器配置应用到集群，请输入以下命令：

```
$ for role in master worker; do
  oc apply -f 99-ipsec-${role}-endpoint-config.yaml
done
```

重要

当 Machine Config Operator (MCO) 更新每个机器配置池中的机器时，它会逐一重启每个节点。在外部 IPsec 连接可用前，您必须等待所有节点更新。

验证

1. 输入以下命令检查机器配置池状态：

```
$ oc get mcp
```

成功更新的节点具有以下状态： **UPDATED=true**、**UPDATING=false**、**DEGRADED=false**。

注意

默认情况下，MCO 会一次在一个池中更新一个机器，从而导致迁移总时间随着集群大小的增加而增加。

2. 要确认 IPsec 机器配置已被成功推出，请输入以下命令：

   a. 确认创建 IPsec 机器配置：

   ```
   $ oc get mc | grep ipsec
   ```

   输出示例

   ```
   80-ipsec-master-extensions        3.2.0        6d15h
   80-ipsec-worker-extensions        3.2.0        6d15h
   ```

   b. 确认已将 IPsec 扩展应用到 control plane 节点：

   ```
   $ oc get mcp master -o yaml | grep 80-ipsec-master-extensions -c
   ```

   c. 确认计算节点的 IPsec 扩展的应用程序。示例输出将显示 **2**。

   ```
   $ oc get mcp worker -o yaml | grep 80-ipsec-worker-extensions -c
   ```

6.8. 其他资源

- IPsec 加密

6.9. 为外部 IPSEC 端点禁用 IPSEC 加密

作为集群管理员，您可以删除外部主机的现有 IPsec 隧道。

先决条件

- 安装 OpenShift CLI (**oc**) 。
- 以具有 **cluster-admin** 权限的用户身份登录集群。
- 您在集群中的 **Full** 或 **External** 模式中启用了 IPsec。

流程

1. 使用以下 YAML 创建名为 **remove-ipsec-tunnel.yaml** 的文件：

```
kind: NodeNetworkConfigurationPolicy
apiVersion: nmstate.io/v1
metadata:
  name: <name>
spec:
  nodeSelector:
    kubernetes.io/hostname: <node_name>
  desiredState:
    interfaces:
    - name: <tunnel_name>
      type: ipsec
      state: absent
```

其中：

name
指定节点网络配置策略的名称。

node_name
指定您要删除的 IPsec 隧道的节点名称。

tunnel_name
指定现有 IPsec 隧道的接口名称。

2. 要删除 IPsec 隧道，请输入以下命令：

```
$ oc apply -f remove-ipsec-tunnel.yaml
```

6.10. 禁用 IPSEC 加密

作为集群管理员，您可以禁用 IPsec 加密。

先决条件

- 已安装 OpenShift CLI（**oc**）。

- 您可以使用具有 **cluster-admin** 权限的用户登陆到集群。

流程

1. 选择以下选项之一来禁用 IPsec 加密：

   a. 其中 **ipsecConfig.mode** 参数设置为 **External** 或 **Full**，**ipsecConfig.full** 模式没有添加到 **networks.operator.openshift.io** 中，请输入以下命令：

   ```
   $ oc patch networks.operator.openshift.io cluster --type=merge -p \
    '{
    "spec":{
      "defaultNetwork":{
        "ovnKubernetesConfig":{
          "ipsecConfig":{
            "mode":"Disabled"
          }}}}}'
   ```

   b. 其中 **ipsecConfig.mode** 参数设置为 **Full**，**ipsecConfig.full** 配置被添加到 **networks.operator.openshift.io**，请输入以下命令：

   ```
   $ oc patch networks.operator.openshift.io cluster --type='json' -p \
         '[{"op": "remove", "path":
   "/spec/defaultNetwork/ovnKubernetesConfig/ipsecConfig/full"},
         {"op": "replace", "path":
   "/spec/defaultNetwork/ovnKubernetesConfig/ipsecConfig/mode", "value": "Disabled"}]'
   ```

2. 可选：您可以将集群 MTU 的大小增加 **46** 字节，因为 IP 数据包中不再有 IPsec 封装安全 Payload (ESP)标头的开销。

6.11. 其他资源

- 在 Red Hat Enterprise Linux (RHEL) 10 中配置使用 IPsec 的 VPN
- 安装 Butane
- 关于 OVN-Kubernetes Container Network Interface (CNI) 网络插件
- 更改集群网络的 MTU
- Network [operator.openshift.io/v1] API

第 7 章 零信任网络

零信任（Zero trust）一个设计安全基础架构的方法，它基于内部的环境，其中的每个交流行为都从一个不受信任的状态开始。这与传统的架构不同。传统架构可能会根据交流是否是在防火墙内发起的来确定其信任性。更具体地说，零信任会尝试缩小安全基础架构中的漏洞（依赖隐式信任模型和一次性身份验证）。

OpenShift Container Platform 可以为平台上运行的容器添加一些零信任网络功能，而无需更改容器或它们中运行的软件。红帽提供的几种产品进一步增加了容器的零信任网络功能。如果您能够更改容器中运行的软件，则还有红帽支持的其他项目，可以添加进一步的功能。

探索零信任网络的以下目标功能。

7.1. 信任的根

公共证书和私钥对于零信任网络至关重要。它们用来识别另一个组件、验证和保护流量的安全组件。证书由其他证书签名，存在到根证书颁发机构 (CA) 的信任链。参与网络的所有内容需要最终具有 root CA 的公钥，以便它可以验证信任链。对于面向公共的项，通常是全局已知的根 CA，其密钥与操作系统、Web 浏览器等一起分发。但是，如果私有 CA 的证书被分发到所有方，则可以针对一个集群或一个机构运行私有 CA。

利用：

- OpenShift Container Platform：OpenShift 在安装时会创建一个集群 CA，用于保护集群资源。但是，OpenShift Container Platform 也可以在集群中为服务证书创建和签署证书，并在请求时可以将集群 CA 捆绑包注入 pod。由 OpenShift Container Platform 创建并签名的服务证书具有 26 个月的生存时间 (TTL)，并在 13 个月内自动轮转。如有必要，也可以手动轮转它们。

- OpenShift cert-manager Operator ：cert-manager 允许您请求由外部信任根签名的密钥。有许多可配置的签发者可以与外部签发者集成，以及与委派的签名证书一起运行的方法。cert-manager API 可供零信任网络中的其他软件使用，以请求必要的证书（如 Red Hat OpenShift Service Mesh），或者可由客户软件直接使用。

7.2. 流量身份验证和加密

确保网络上的所有流量都加密，并且端点是可识别的。例如，Mutual TLS 或 mTLS，这是 mutual 身份验证的方法。

利用：

- OpenShift Container Platform：使用透明 pod-to-pod IPsec，可以通过 IP 地址识别流量的源和目标。这可以实现对出口流量使用 IPsec 加密。通过使用 egress IP 功能，流量的源 IP 地址可用于识别集群中的流量源。

- Red Hat OpenShift Service Mesh：提供强大的 mTLS 功能，这些功能可以透明地增加离开 pod 的流量以提供身份验证和加密。

- OpenShift cert-manager Operator ：使用自定义资源定义 (CRD) 请求可挂载用于 SSL/TLS 协议的证书。

7.3. 身份识别和验证

在使用 CA 最小化证书后，您可以通过验证连接端的另一端的连接身份验证一个用户或客户端机器来建立信任关系。这还需要管理证书生命周期，以便在被破坏时限制使用。

利用：

- OpenShift Container Platform：集群签名的服务证书，以确保客户端与可信端点通信。这要求服务使用 SSL/TLS，并且客户端使用集群 CA。客户端身份必须使用某种其他方法提供。
- Red Hat Single Sign-On ：提供与企业用户目录或第三方身份提供程序的请求身份验证集成。
- Red Hat OpenShift Service Mesh：将连接透明升级到 mTLS、自动轮转、自定义证书过期以及请求使用 JSON Web 令牌 (JWT) 的身份验证。
- OpenShift cert-manager Operator ：创建和管理证书，供应用程序使用。证书可以由 CRD 控制并挂载为 secret，也可以更改应用程序以直接与 cert-manager API 交互。

7.4. 服务间的授权

务必要根据请求者的身份控制对服务的访问。这由平台完成，不需要每个应用程序来实现它。这样可以更好地审核和检查策略。

利用：

- OpenShift Container Platform：可以使用 Kubernetes **NetworkPolicy** 和 **AdminNetworkPolicy** 对象在平台的网络层中强制实施隔离。
- Red Hat OpenShift Service Mesh：使用标准 Istio 对象 对流量的复杂 L4 和 L7 控制，并使用 mTLS 识别流量源和目标，然后根据该信息应用策略。

7.5. 事务级验证

除了识别和验证连接的功能外，控制单个事务的访问也很有用。这可包括源、可观察性和语义验证的速率限制。

利用：

- Red Hat OpenShift Service Mesh：执行对请求的 L7 检查，拒绝 HTTP 请求、事务级 可观察性和报告。Service Mesh 还可使用 JWT 提供基于请求的身份验证。

7.6. 风险评估

随着集群中的安全策略数量增加，策略允许和拒绝的视觉化变得越来越重要。这些工具可让您更轻松地创建、视觉化和管理集群安全策略。

利用：

- Red Hat OpenShift Service Mesh：使用 OpenShift Web 控制台创建和视觉化 Kubernetes **NetworkPolicy** 和 **AdminNetworkPolicy**，以及 OpenShift Networking **EgressFirewall** 对象。
- Red Hat Advanced Cluster Security for Kubernetes ：对象的高级视觉化。

7.7. 站点范围内的策略实施和分发

在集群中部署应用程序后，管理组成安全规则的所有对象就变得困难。务必要应用站点范围的策略，并审核部署的对象是否符合策略。这应该允许将某些权限委派给定义的绑定内的用户和集群管理员，并应允许例外策略。

利用：

- Red Hat OpenShift Service Mesh : RBAC 来控制策略对象并委派控制。
- Red Hat Advanced Cluster Security for Kubernetes ：策略实施 引擎。
- Red Hat Advanced Cluster Management (RHACM) for Kubernetes : 集中式策略控制。

7.8. 持续的可观察性，以及重新检查评估

运行的集群后，您希望能够观察流量，并使用定义的规则验证流量是否被端口。这对入侵检测、共识非常重要，有助于操作负载管理。

利用：

- Network Observability Operator ：允许检查、监控和警报到集群中的 pod 和节点的网络连接。
- Red Hat Advanced Cluster Management (RHACM) for Kubernetes : 监控、收集和评估系统级事件，如进程执行、网络连接和流以及特权升级。它可以决定集群的基线，然后检测异常活动并提醒您。
- Red Hat OpenShift Service Mesh ： 可以监控进入和离开 pod 的流量。
- Red Hat OpenShift distributed tracing 平台 ：对于适当的检测应用程序，您可以看到与特定操作关联的所有流量，因为它被分成到微服务的子请求。这可让您识别分布式应用程序中的瓶颈。

7.9. 端点安全性

务必要信任在集群中运行服务的软件没有被破坏。例如，您可能需要确保在可信硬件上运行认证的镜像，并有策略只允许基于端点特征或来自端点特征的连接。

利用：

- OpenShift Container Platform：Secureboot 可以确保集群中的节点正在运行可信软件，因此平台本身（包括容器运行时）没有被篡改。您可以将 OpenShift Container Platform 配置为仅运行由特定签名签名的镜像。
- Red Hat Trusted Artifact Signer ：这可用于可信构建链并生成已签名的容器镜像。

7.10. 在集群外扩展信任

您可能希望通过允许集群到子域的 mint CA 来在集群外扩展信任。或者，您可能希望 attest 到集群中的工作负载身份到远程端点。

利用：

- OpenShift cert-manager Operator ：您可以使用 cert-manager 管理委派的 CA，以便可以在不同集群之间或通过您的机构分发信任。
- Red Hat OpenShift Service Mesh : 可以使用 SPIFFE 为在远程或本地集群中运行的端点提供远程测试工作负载。