OpenShift Container Platform 4.19

配置网络设置

OpenShift Container Platform 中的常规网络配置过程

Last Updated: 2025-09-11

OpenShift Container Platform 4.19 配置网络设置

OpenShift Container Platform 中的常规网络配置过程

Legal Notice

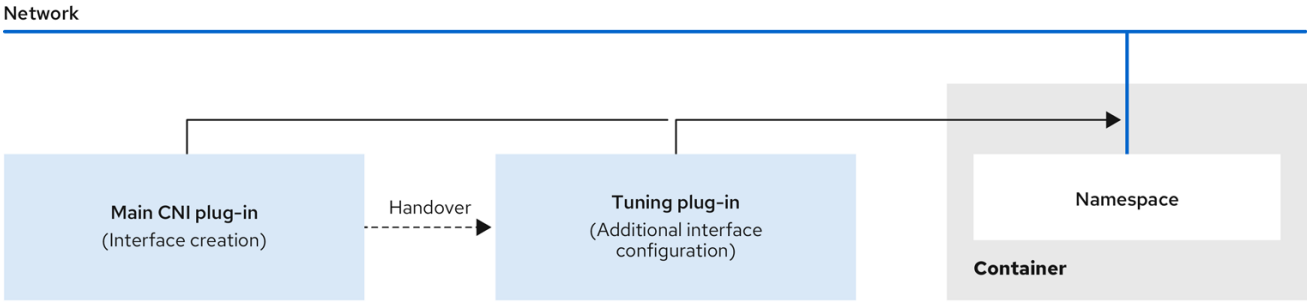
Abstract

本文档论述了在 OpenShift Container Platform 中配置网络方面，如节点端口服务、IP 地址范围、IP 故障转移和集群范围的代理。

Table of Contents

第 1 章 使用调优插件配置系统控制和接口属性

在 Linux 中，管理员可通过 sysctl 在运行时修改内核参数。您可以使用调优 Container Network Interface(CNI)元插件修改接口级网络 sysctl。tuning CNI meta 插件在一个链中运行，主 CNI 插件如下所示。



主 CNI 插件分配接口，并将此接口在运行时传递给 tuning CNI meta 插件。您可以使用 tuning CNI meta 插件更改网络命名空间中的一些 sysctl 和几个接口属性，如 promiscuous 模式、all-multicast 模式、MTU 和 MAC 地址。

1.1. 使用 TUNING CNI 配置系统控制

以下流程将调整 CNI 配置为更改接口级网络 **net.ipv4.conf.IFNAME.accept_redirects** sysctl。这个示例启用接受和发送 ICMP 重定向的数据包。在 tuning CNI meta 插件配置中，接口名称由 **IFNAME** 令牌表示，并替换为运行时接口的实际名称。

流程

1. 使用以下内容创建网络附加定义，如 **tuning-example.yaml** ：

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: <name> 1
  namespace: default 2
spec:
  config: '{
    "cniVersion": "0.4.0", 3
    "name": "<name>", 4
    "plugins": [{
      "type": "<main_CNI_plugin>" 5
     },
     {
      "type": "tuning", 6
      "sysctl": {
          "net.ipv4.conf.IFNAME.accept_redirects": "1" 7
      }
     }
    ]
}
```

1. 指定要创建的额外网络附加的名称。名称在指定的命名空间中必须是唯一的。
2. 指定与对象关联的命名空间。
3. 指定 CNI 规格版本。
4. 指定配置的名称。建议您将配置名称与网络附加定义的 name 值匹配。
5. 指定要配置的主 CNI 插件的名称。
6. 指定 CNI meta 插件的名称。
7. 指定要设置的 sysctl。接口名称由 **IFNAME** 令牌表示，并替换为运行时接口的实际名称。

下面显示了一个 YAML 文件示例：

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: tuningnad
  namespace: default
spec:
  config: '{
    "cniVersion": "0.4.0",
    "name": "tuningnad",
    "plugins": [{
      "type": "bridge"
    },
    {
      "type": "tuning",
      "sysctl": {
        "net.ipv4.conf.IFNAME.accept_redirects": "1"
      }
    }
  ]
}'
```

2. 运行以下命令来应用 YAML：

```
$ oc apply -f tuning-example.yaml
```

输出示例

```
networkattachmentdefinition.k8.cni.cncf.io/tuningnad created
```

3. 使用类似以下示例的网络附加定义，创建示例 **pod.yaml** ：

```
apiVersion: v1
kind: Pod
metadata:
  name: tunepod
  namespace: default
  annotations:
    k8s.v1.cni.cncf.io/networks: tuningnad 1
```

```
spec:
  containers:
  - name: podexample
    image: centos
    command: ["/bin/bash", "-c", "sleep INF"]
    securityContext:
      runAsUser: 2000 2
      runAsGroup: 3000 3
      allowPrivilegeEscalation: false 4
      capabilities: 5
        drop: ["ALL"]
  securityContext:
    runAsNonRoot: true 6
    seccompProfile: 7
      type: RuntimeDefault
```

1. 指定配置的 **NetworkAttachmentDefinition** 的名称。
2. **runAsUser** 控制使用哪个用户 ID 运行容器。
3. **runAsGroup** 控制容器使用哪个主要组 ID。
4. **allowPrivilegeEscalation** 决定 pod 是否请求允许特权升级。如果未指定，则默认为 true。这个布尔值直接控制在容器进程中是否设置了 **no_new_privs** 标志。
5. **capabilities** 允许特权操作，而不提供完整的 root 访问权限。此策略可确保从 pod 中丢弃了所有功能。
6. **runAsNonRoot: true** 要求容器使用 0 以外的任何 UID 运行。
7. **RuntimeDefault** 为 pod 或容器工作负载启用默认的 seccomp 配置集。

4. 运行以下命令来应用 yaml:

```
$ oc apply -f examplepod.yaml
```

5. 运行以下命令验证 pod 是否已创建：

```
$ oc get pod
```

输出示例

```
NAME     READY   STATUS    RESTARTS   AGE
tunepod  1/1     Running   0          47s
```

6. 运行以下命令登录到 pod：

```
$ oc rsh tunepod
```

7. 验证配置的 sysctl 标记的值。例如，通过运行以下命令查找 **net.ipv4.conf.net1.accept_redirects** 的值：

```
sh-4.4# sysctl net.ipv4.conf.net1.accept_redirects
```

预期输出

```
net.ipv4.conf.net1.accept_redirects = 1
```

1.2. 使用调优 CNI 启用 ALL-MULTICAST 模式

您可以使用 tuning Container Network Interface (CNI) meta 插件启用 all-multicast 模式。

以下流程描述了如何配置调优 CNI 来启用 all-multicast 模式。

流程

1. 使用以下内容创建网络附加定义，如 **tuning-example.yaml** ：

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: <name> 1
  namespace: default 2
spec:
  config: '{
    "cniVersion": "0.4.0", 3
    "name": "<name>", 4
    "plugins": [{
      "type": "<main_CNI_plugin>" 5
    },
    {
      "type": "tuning", 6
      "allmulti": true 7
    }
   ]
}
```

1. 指定要创建的额外网络附加的名称。名称在指定的命名空间中必须是唯一的。
2. 指定与对象关联的命名空间。
3. 指定 CNI 规格版本。
4. 指定配置的名称。将配置名称与网络附加定义的 name 值匹配。
5. 指定要配置的主 CNI 插件的名称。
6. 指定 CNI meta 插件的名称。
7. 更改接口的 all-multicast 模式。如果启用，接口将接收网络上的所有多播数据包。

下面显示了一个 YAML 文件示例：

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: setallmulti
  namespace: default
spec:
  config: '{
    "cniVersion": "0.4.0",
    "name": "setallmulti",
    "plugins": [
      {
        "type": "bridge"
      },
      {
        "type": "tuning",
        "allmulti": true
      }
    ]
  }'
```

2. 运行以下命令应用 YAML 文件中指定的设置：

```
$ oc apply -f tuning-allmulti.yaml
```

输出示例

```
networkattachmentdefinition.k8s.cni.cncf.io/setallmulti created
```

3. 使用类似以下 **examplepod.yaml**文件中指定的网络附加定义创建 pod ：

```
apiVersion: v1
kind: Pod
metadata:
  name: allmultipod
  namespace: default
  annotations:
    k8s.v1.cni.cncf.io/networks: setallmulti (1)
spec:
  containers:
  - name: podexample
    image: centos
    command: ["/bin/bash", "-c", "sleep INF"]
    securityContext:
      runAsUser: 2000 (2)
      runAsGroup: 3000 (3)
      allowPrivilegeEscalation: false (4)
      capabilities: (5)
        drop: ["ALL"]
  securityContext:
    runAsNonRoot: true (6)
    seccompProfile: (7)
      type: RuntimeDefault
```

1. 指定配置的 **NetworkAttachmentDefinition** 的名称。
2. 指定运行容器的用户 ID。
3. 指定容器使用哪个主要组 ID。
4. 指定 pod 是否可以请求特权升级。如果未指定，则默认为 **true**。这个布尔值直接控制在容器进程中是否设置了 **no_new_privs** 标志。
5. 指定容器功能。**drop: ["ALL"]** 语句表示所有 Linux 功能都会从 pod 中丢弃，提供更严格的安全配置集。
6. 指定容器将使用任何 UID 为 0 的用户运行。
7. 指定容器的 seccomp 配置集。在这种情况下，type 被设置为 **RuntimeDefault**。seccomp 是一个 Linux 内核功能，它限制了进程可用的系统调用，通过最小化攻击面来提高安全性。

4. 运行以下命令应用 YAML 文件中指定的设置：

```
$ oc apply -f examplepod.yaml
```

5. 运行以下命令验证 pod 是否已创建：

```
$ oc get pod
```

输出示例

```
NAME        READY   STATUS    RESTARTS   AGE
allmultipod  1/1     Running   0          23s
```

6. 运行以下命令登录到 pod：

```
$ oc rsh allmultipod
```

7. 运行以下命令，列出与 pod 关联的所有接口：

```
sh-4.4# ip link
```

输出示例

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode
DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0@if22: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8901 qdisc noqueue state
UP mode DEFAULT group default
    link/ether 0a:58:0a:83:00:10 brd ff:ff:ff:ff:ff:ff link-netnsid 0 (1)
3: net1@if24: <BROADCAST,MULTICAST,ALLMULTI,UP,LOWER_UP> mtu 1500 qdisc
noqueue state UP mode DEFAULT group default
    link/ether ee:9b:66:a4:ec:1d brd ff:ff:ff:ff:ff:ff link-netnsid 0 (2)
```

1. **eth0@if22** 是主接口

2 **net1@if24** 是配置了 network-attachment-definition 的二级接口，它支持 all-multicast 模式 (ALLMULTI 标志)

1.3. 其他资源

- 在容器中使用 sysctl
- SR-IOV 网络节点配置对象
- 为 SR-IOV 网络配置接口级网络 sysctl 设置和 all-multicast 模式

第 2 章 配置节点端口服务范围

在集群安装过程中，您可以配置节点端口范围来满足集群的要求。在集群安装后，只有集群管理员可以将范围扩展为安装后任务。如果您的集群使用大量节点端口，请考虑根据集群的要求增加可用端口范围。

如果您没有在集群安装过程中设置节点端口范围，则默认范围 **30000-32768** 应用到您的集群。在这种情况下，您可以在任一端扩展范围，但您必须在新端口范围中保留 **30000-32768**。

重要

红帽没有在默认的端口范围 **30000-32768** 之外执行测试。对于默认端口范围以外的范围，请确保测试以验证扩展节点端口范围不会影响您的集群。特别是，请确保存在：

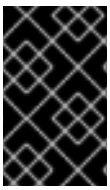
- 不与主机进程已经使用的任何端口重叠
- 没有与已经由使用主机网络配置的 pod 使用的端口重叠

如果您扩展范围和端口分配问题，请创建新集群并为其设置所需的范围。

如果扩展节点端口范围和 OpenShift CLI (**oc**)会停止工作，因为端口与 OpenShift Container Platform API 服务器冲突，您必须创建新集群。

2.1. 扩展节点端口范围

您可以扩展集群的节点端口范围。安装 OpenShift Container Platform 集群后，您无法在当前配置的范围内缩小节点端口范围。



重要

红帽没有在默认的端口范围 **30000-32768** 之外执行测试。对于默认端口范围以外的范围，请确保测试以验证扩展节点端口范围是否不会影响您的集群。如果您扩展范围和端口分配问题，请创建新集群并为其设置所需的范围。

先决条件

- 已安装 OpenShift CLI (**oc**)。
- 以具有 **cluster-admin** 权限的用户身份登录集群。
- 您确保集群基础架构允许访问扩展范围中存在的端口。例如，如果您将节点端口范围扩展到 **30000-32900**，您的防火墙或数据包过滤配置必须允许包含端口范围 **30000-32900**。

流程

- 要扩展集群用来管理 pod 流量的 **network.config.openshift.io** 对象中的 **serviceNodePortRange** 参数的范围，请输入以下命令：

```
$ oc patch network.config.openshift.io cluster --type=merge -p \
  '{
    "spec":
      { "serviceNodePortRange": "<port_range>" }
  }'
```

其中：

<port_range>

指定您扩展的范围，如 **30000-32900**。

提示

您还可以应用以下 YAML 来更新节点端口范围：

```
apiVersion: config.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  serviceNodePortRange: "<port_range>"
# ...
```

输出示例

```
network.config.openshift.io/cluster patched
```

验证

- 要确认更新的配置处于活跃状态，请输入以下命令。应用更新可能需要几分钟时间。

```
$ oc get configmaps -n openshift-kube-apiserver config \
  -o jsonpath="{.data['config\.yaml']}" | \
  grep -Eo '"service-node-port-range":["[[:digit:]]+-[[:digit:]]+"]'
```

输出示例

```
"service-node-port-range":["30000-32900"]
```

2.2. 其他资源

- 使用 NodePort 配置集群入口流量
- Network [config.openshift.io/v1]
- Service [core/v1]

第 3 章 配置集群网络范围

作为集群管理员，您可以在集群安装后扩展集群网络范围。如果额外的节点需要更多 IP 地址，您可能需要扩展集群网络范围。

例如，如果您部署了集群，并将 **10.128.0.0/19** 指定为集群网络范围，主机前缀为 **23**，则限制为 16 个节点。您可以通过将集群中的 CIDR 掩码更改为 **/14** 来扩展到 510 个节点。

在扩展集群网络地址范围时，集群必须使用 OVN-Kubernetes 网络插件。不支持其他网络插件。

修改集群网络 IP 地址范围时会有以下限制：

- 指定的 CIDR 掩码大小必须总是小于当前配置的 CIDR 掩码大小，因为您只能向已安装的集群添加更多节点来增加 IP 空间
- 无法修改主机前缀
- 使用覆盖默认网关配置的 Pod 必须在集群网络扩展后重新创建

3.1. 扩展集群网络 IP 地址范围

您可以扩展集群网络的 IP 地址范围。由于这个更改需要在集群中推出新的 Operator 配置，所以最多可能需要 30 分钟才能生效。

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。
- 确保集群使用 OVN-Kubernetes 网络插件。

流程

1. 要获取集群的集群网络范围和主机前缀，请输入以下命令：

```
$ oc get network.operator.openshift.io \
  -o jsonpath="{.items[0].spec.clusterNetwork}"
```

输出示例

```
[{"cidr":"10.217.0.0/22","hostPrefix":23}]
```

2. 要扩展集群网络 IP 地址范围，请输入以下命令。使用上一命令输出返回的 CIDR IP 地址范围和主机前缀。

```
$ oc patch Network.config.openshift.io cluster --type='merge' --patch \
  '{
    "spec":{
      "clusterNetwork": [ {"cidr":"<network>/<cidr>","hostPrefix":<prefix>} ],
      "networkType": "OVNKubernetes"
    }
  }'
```

其中：

<network>

指定您在上一步中获取的 **cidr** 字段的网络部分。您无法更改这个值。

<cidr>

指定网络前缀长度。例如，**14**。将此值更改为比上一步中的输出值小的值，以扩展集群网络范围。

<prefix>

指定集群的当前主机前缀。这个值必须与您在上一步中获取的 **hostPrefix** 字段的值相同。

示例命令

```
$ oc patch Network.config.openshift.io cluster --type='merge' --patch \
  '{
    "spec":{
      "clusterNetwork": [ {"cidr":"10.217.0.0/14","hostPrefix": 23} ],
      "networkType": "OVNKubernetes"
    }
  }'
```

输出示例

```
network.config.openshift.io/cluster patched
```

3. 要确认配置是活跃的，请输入以下命令。可能需要 30 分钟才能使此更改生效。

```
$ oc get network.operator.openshift.io \
  -o jsonpath="{.items[0].spec.clusterNetwork}"
```

输出示例

```
[{"cidr":"10.217.0.0/14","hostPrefix":23}]
```

3.2. 其他资源

- Red Hat OpenShift Network Calculator
- 关于 OVN-Kubernetes 网络插件

第 4 章 配置 IP 故障转移

本节论述了为 OpenShift Container Platform 集群上的 pod 和服务配置 IP 故障转移。

IP 故障转移使用 Keepalived 在一组主机上托管一组外部访问的虚拟 IP (VIP) 地址。每个 VIP 地址仅由单个主机提供服务。Keepalived 使用虚拟路由器冗余协议（VRRP）决定在主机集合中使用哪个主机提供 VIP 服务。如果主机不可用，或者 Keepalived 正在监视的服务没有响应，则 VIP 会切换到主机集中的另外一个主机。这意味着只要主机可用，便始终可以提供 VIP 服务。

集合中的每个 VIP 都由从集合中选择的节点提供服务。如果单个节点可用，则会提供 VIP。无法将 VIP 显式分发到节点上，因此可能存在没有 VIP 的节点和其他具有多个 VIP 的节点。如果只有一个节点，则所有 VIP 都在其中。

管理员必须确保所有 VIP 地址都满足以下要求：

- 可在集群外部配置的主机上访问。
- 不用于集群中的任何其他目的。

每个节点上的 keepalived 确定所需服务是否在运行。如果是，则支持 VIP，Keepalived 参与协商来确定哪个节点服务 VIP。对于要参与的节点，服务必须侦听 VIP 上的观察端口，或者必须禁用检查。

注意

集合中的每个 VIP 都可以由不同的节点提供。

IP 故障转移会监控每个 VIP 上的端口，以确定该端口能否在节点上访问。如果端口无法访问，则不会向节点分配 VIP。如果端口设为 **0**，则会禁止此检查。检查脚本执行所需的测试。

当运行 Keepalived 的节点通过检查脚本时，该节点上的 VIP 可以根据其优先级和当前 master 的优先级以及抢占策略决定进入 **master** 状态。

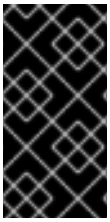
集群管理员可以通过 **OPENSHIFT_HA_NOTIFY_SCRIPT** 变量提供一个脚本，每当节点上的 VIP 的状态发生变化时会调用此脚本。keepalived 在为 VIP 提供服务时为 **master** 状态；当另一个节点提供 VIP 服务时，状态为 **backup**；当检查脚本失败时，状态为 **fault**。每当状态更改时，notify 脚本都会被调用，并显示新的状态。

您可以在 OpenShift Container Platform 上创建 IP 故障转移部署配置。IP 故障转移部署配置指定 VIP 地址的集合，以及服务它们的一组节点。一个集群可以具有多个 IP 故障转移部署配置，各自管理自己的一组唯一的 VIP 地址。IP 故障转移配置中的每个节点运行 IP 故障转移 pod，此 pod 运行 Keepalived。

使用 VIP 访问带有主机网络的 pod 时，应用程序 pod 在运行 IP 故障转移 pod 的所有节点上运行。这可让任何 IP 故障转移节点成为主节点，并在需要时为 VIP 服务。如果应用程序 pod 没有在所有具有 IP 故障转移功能的节点上运行，有些 IP 故障转移节点不会为 VIP 服务，或者某些应用 pod 都不会接收任何流量。对 IP 故障转移和应用容器集使用相同的选择器和复制数，以避免这种不匹配。

在使用 VIP 访问服务时，任何节点都可以位于节点的 IP 故障转移集中，因为无论应用容器集在哪里运行，该服务都可以在所有节点上访问。任何 IP 故障转移节点可以随时变成主节点。服务可以使用外部 IP 和服务端口，或者可以使用 **NodePort**。设置 **NodePort** 是一个特权操作。

在服务定义中使用外部 IP 时，VIP 被设置为外部 IP，IP 故障转移监控端口则设为服务端口。在使用节点端口时，该端口在集群的每个节点上打开，服务则从当前服务于 VIP 的任何节点对流量进行负载平衡。在这种情况下，IP 故障转移监控端口在服务定义中设置为 **NodePort**。



重要

即使一个服务 VIP 具有高可用性，但性能仍会受到影响。keepalived 确保每个 VIP 都由配置中的某个节点提供服务，即使其他节点没有，也可以在同一节点上出现多个 VIP。当 IP 故障转移在同一节点上放置多个 VIP 时，在一组 VIP 间进行外部负载平衡的策略可能会被破解。

当使用 **ExternalIP** 时，您可以将 IP 故障转移设置为与 **ExternalIP** 范围相同的 VIP 范围。您还可以禁用监控端口。在这种情况下，所有 VIP 都出现在集群中的同一节点上。任何用户都可以使用 **ExternalIP** 设置服务并使其高度可用。

重要

集群中最多有 254 个 VIP。

4.1. IP 故障转移环境变量

下表包含用于配置 IP 故障转移的变量。

表 4.1. IP 故障转移环境变量

变量名称	default	描述
OPENSHIFT_HA_MONITOR_PORT	**80**	IP 故障转移 pod 会尝试在每个虚拟 IP（VIP）上打开到此端口的 TCP 连接。如果建立连接，则服务将被视为正在运行。如果此端口设为 **0**，则测试会始终通过。
OPENSHIFT_HA_NETWORK_INTERFACE		IP 故障转移用于发送虚拟路由器冗余协议 (VRRP) 流量的接口名称。默认值为 **eth0**。 如果您的集群使用 OVN-Kubernetes 网络插件，请将此值设置为 **br-ex** 以避免数据包丢失。对于使用 OVN-Kubernetes 网络插件的集群，所有侦听接口都不会为 VRRP 提供服务，而是预期入站流量会通过 **br-ex** 网桥进行。
OPENSHIFT_HA_REPLICA_COUNT	**2**	要创建的副本数。这必须与 IP 故障转移部署配置中的 **spec.replicas** 值匹配。
OPENSHIFT_HA_VIRTUAL_IPS		要复制的 IP 地址范围列表。必须提供.例如，**1.2.3.4-6,1.2.3.9**。
OPENSHIFT_HA_VRRP_ID_OFFSET	**10**	用于设置虚拟路由器 ID 的偏移值。使用不同的偏移值可以在同一集群中存在多个 IP 故障转移配置。默认偏移值为 **10**，允许的范围为 **0** 到 **255**。
OPENSHIFT_HA_VIP_GROUPS		为 VRRP 创建的组数量。如果没有设置，则会为通过 **OPENSHIFT_HA_VIP_GROUPS** 变量指定的每个虚拟 IP 范围创建一个组。

变量名称	default	描述
OPENSHIFT_HA_IPTABLES_CHAIN	输入	iptables 链的名称，用于自动添加允许 VRRP 流量的 **iptables** 规则。如果没有设置值，则不会添加 **iptables** 规则。如果链不存在，则不会创建它。
OPENSHIFT_HA_CHECK_SCRIPT		定期运行的脚本的 pod 文件系统中的完整路径名称，以验证应用是否正在运行。
OPENSHIFT_HA_CHECK_INTERVAL	**2**	检查脚本运行的期间（以秒为单位）。
OPENSHIFT_HA_NOTIFY_SCRIPT		当状态发生变化时运行的脚本的 pod 文件系统的完整路径名称。
OPENSHIFT_HA_PREEMPTION	**preempt_nodelay 300**	处理新的具有更高优先级主机的策略。**nopreempt** 策略不会将 master 从较低优先级主机移到优先级更高的主机。

4.2. 在集群中配置 IP 故障切换

作为集群管理员，您可以在整个集群中或在其中的一部分节点（由标签选项器定义）中配置 IP 故障转移。您还可以在集群中配置多个 IP 故障转移部署，每个 IP 故障转移部署相互独立。

IP 故障转移部署确保故障转移 pod 在符合限制或使用的标签的每个节点上运行。

此 pod 运行 Keepalived，它可以监控端点，并在第一个节点无法访问服务或端点时使用 Virtual Router Redundancy Protocol（VRRP）从一个节点切换到另一个节点的虚拟 IP（VIP）。

对于生产环境，设置一个**选择器（selector）**，用于选择至少两个节点，并设置与所选节点数量相等的**副本**。

先决条件

- 以具有 **cluster-admin** 权限的用户身份登录集群。
- 已创建一个 pull secret。
- 仅限 Red Hat OpenStack Platform (RHOSP)：
  - 您在目标环境中安装了 RHOSP 客户端 (RHCOS 文档)。
  - 您还下载了 RHOSP **openrc.sh** rc 文件 (RHCOS 文档)。

流程

1. 创建 IP 故障转移服务帐户：

```
$ oc create sa ipfailover
```

2. 为 **hostNetwork** 更新安全性上下文约束（SCC）：

```
$ oc adm policy add-scc-to-user privileged -z ipfailover
```

```
$ oc adm policy add-scc-to-user hostnetwork -z ipfailover
```

3. 仅限 Red Hat OpenStack Platform (RHOSP)：完成以下步骤，使在 RHOSP 端口上可以访问故障转移 VIP 地址。

   a. 使用 RHOSP CLI 在 RHOSP 集群的 **allowed_address_pairs** 参数中显示默认的 RHOSP API 和 VIP 地址：

   ```
   $ openstack port show <cluster_name> -c allowed_address_pairs
   ```

   输出示例

   ```
   *Field*                 *Value*
   allowed_address_pairs    ip_address='192.168.0.5', mac_address='fa:16:3e:31:f9:cb'
                            ip_address='192.168.0.7', mac_address='fa:16:3e:31:f9:cb'
   ```

   b. 为 IP 故障转移部署设置不同的 VIP 地址，并通过在 RHOSP CLI 中输入以下命令使 RHOSP 端口上的地址访问。不要将任何默认的 RHOSP API 和 VIP 地址设置为 IP 故障转移部署的故障转移 VIP 地址。

   在 RHOSP 端口中添加 1.1.1.1 故障转移 IP 地址作为允许的地址的示例。

   ```
   $ openstack port set <cluster_name> --allowed-address ip-address=1.1.1.1,mac-address=fa:fa:16:3e:31:f9:cb
   ```

   c. 创建部署 YAML 文件，为您的部署配置 IP 故障切换。请参阅后续步骤中的"IP 故障转移配置的部署 YAML 示例"。

   d. 在 IP 故障转移部署中指定以下规格，以便将故障转移 VIP 地址传递给 **OPENSHIFT_HA_VIRTUAL_IPS** 环境变量：

   将 1.1.1.1 VIP 地址添加到 OPENSHIFT_HA_VIRTUAL_IPS 的示例

   ```
   apiVersion: apps/v1
   kind: Deployment
   metadata:
     name: ipfailover-keepalived
   # ...
       spec:
         env:
         - name: OPENSHIFT_HA_VIRTUAL_IPS
         value: "1.1.1.1"
   # ...
   ```

4. 创建部署 YAML 文件来配置 IP 故障切换。

注意

对于 Red Hat OpenStack Platform (RHOSP)，您不需要重新创建部署 YAML 文件。您已作为之前说明的一部分创建了此文件。

IP 故障转移配置的部署 YAML 示例

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ipfailover-keepalived 1
  labels:
    ipfailover: hello-openshift
spec:
  strategy:
    type: Recreate
  replicas: 2
  selector:
    matchLabels:
      ipfailover: hello-openshift
  template:
    metadata:
      labels:
        ipfailover: hello-openshift
    spec:
      serviceAccountName: ipfailover
      privileged: true
      hostNetwork: true
      nodeSelector:
        node-role.kubernetes.io/worker: ""
      containers:
      - name: openshift-ipfailover
        image: registry.redhat.io/openshift4/ose-keepalived-ipfailover-rhel9:v4.19
        ports:
        - containerPort: 63000
          hostPort: 63000
        imagePullPolicy: IfNotPresent
        securityContext:
          privileged: true
        volumeMounts:
        - name: lib-modules
          mountPath: /lib/modules
          readOnly: true
        - name: host-slash
          mountPath: /host
          readOnly: true
          mountPropagation: HostToContainer
        - name: etc-sysconfig
          mountPath: /etc/sysconfig
          readOnly: true
        - name: config-volume
          mountPath: /etc/keepalive
        env:
        - name: OPENSHIFT_HA_CONFIG_NAME
          value: "ipfailover"
```

```
    - name: OPENSHIFT_HA_VIRTUAL_IPS 2
      value: "1.1.1.1-2"
    - name: OPENSHIFT_HA_VIP_GROUPS 3
      value: "10"
    - name: OPENSHIFT_HA_NETWORK_INTERFACE 4
      value: "ens3" #The host interface to assign the VIPs
    - name: OPENSHIFT_HA_MONITOR_PORT 5
      value: "30060"
    - name: OPENSHIFT_HA_VRRP_ID_OFFSET 6
      value: "10"
    - name: OPENSHIFT_HA_REPLICA_COUNT 7
      value: "2" #Must match the number of replicas in the deployment
    - name: OPENSHIFT_HA_USE_UNICAST
      value: "false"
    #- name: OPENSHIFT_HA_UNICAST_PEERS
      #value: "10.0.148.40,10.0.160.234,10.0.199.110"
    - name: OPENSHIFT_HA_IPTABLES_CHAIN 8
      value: "INPUT"
    #- name: OPENSHIFT_HA_NOTIFY_SCRIPT 9
    #  value: /etc/keepalive/mynotifyscript.sh
    - name: OPENSHIFT_HA_CHECK_SCRIPT 10
      value: "/etc/keepalive/mycheckscript.sh"
    - name: OPENSHIFT_HA_PREEMPTION 11
      value: "preempt_delay 300"
    - name: OPENSHIFT_HA_CHECK_INTERVAL 12
      value: "2"
    livenessProbe:
      initialDelaySeconds: 10
      exec:
        command:
        - pgrep
        - keepalived
  volumes:
  - name: lib-modules
    hostPath:
      path: /lib/modules
  - name: host-slash
    hostPath:
      path: /
  - name: etc-sysconfig
    hostPath:
      path: /etc/sysconfig
  # config-volume contains the check script
  # created with `oc create configmap keepalived-checkscript --from-file=mycheckscript.sh`
  - configMap:
      defaultMode: 0755
      name: keepalived-checkscript
    name: config-volume
  imagePullSecrets:
    - name: openshift-pull-secret 13
```

1 IP 故障转移部署的名称。

2 要复制的 IP 地址范围列表。必须提供.例如，**1.2.3.4-6,1.2.3.9**。

3 为 VRRP 创建的组数量。如果没有设置，则会为通过 **OPENSHIFT_HA_VIP_GROUPS** 变量指定的每个虚拟 IP 范围创建一个组。

4 IP 故障切换用于发送 VRRP 流量的接口名称。默认情况下使用 **eth0**。

5 IP 故障转移 pod 会尝试在每个 VIP 上打开到此端口的 TCP 连接。如果建立连接，则服务将被视为正在运行。如果此端口设为 **0**，则测试会始终通过。默认值为 **80**。

6 用于设置虚拟路由器 ID 的偏移值。使用不同的偏移值可以在同一集群中存在多个 IP 故障转移配置。默认偏移值为 **10**，允许的范围为 **0** 到 **255**。

7 要创建的副本数。这必须与 IP 故障转移部署配置中的 **spec.replicas** 值匹配。默认值为 **2**。

8 **iptables** 链的名称，用于自动添加允许 VRRP 流量的 **iptables** 规则。如果没有设置值，则不会添加 **iptables** 规则。如果链不存在，则不会创建链，Keepalived 在单播模式下运行。默认为 **INPUT**。

9 当状态发生变化时运行的脚本的 pod 文件系统的完整路径名称。

10 定期运行的脚本的 pod 文件系统中的完整路径名称，以验证应用是否正在运行。

11 处理新的具有更高优先级主机的策略。默认值为 **preempt_delay 300**，这会导致，在有一个较低优先级的 master 提供 VIP 时，Keepalived 实例在 5 分钟后会接管 VIP。

12 检查脚本运行的期间（以秒为单位）。默认值为 **2**。

13 在创建部署之前创建 pull secret，否则您将在创建部署时收到错误。

4.3. 配置检查和通知脚本

keepalived 通过定期运行可选用户提供的检查脚本来监控应用程序的健康状况。例如，该脚本可以通过发出请求并验证响应来测试 Web 服务器。作为集群管理员，您可以提供一个可选的 notify 脚本，该脚本会在状态发生变化时调用。

检查和通知在 IP 故障转移容器集中运行的脚本，并使用容器集文件系统，而不是主机文件系统。但是，IP 故障转移 pod 使主机文件系统在 **/hosts** 挂载路径下可用。在配置检查或通知脚本时，您必须提供脚本的完整路径。提供脚本的建议方法是使用 **ConfigMap** 对象。

检查和通知脚本的完整路径名称添加到 Keepalived 配置文件 **_/etc/keepalived/keepalived.conf** 中，该文件会在 Keepalived 每次启动时加载。可以使用 **ConfigMap** 对象将脚本添加到 pod，如以下方法所述。

检查脚本

不提供检查脚本时，将运行一个简单的默认脚本来测试 TCP 连接。当监控端口为 **0** 时，禁止此默认测试。

每个 IP 故障转移 pod 管理一个 Keepalived 守护进程，在运行 pod 的节点上管理一个或多个虚拟 IP (VIP) 地址。Keepalived 守护进程为该节点保留每个 VIP 的状态。特定节点上的特定 VIP 可能处于 **master**、**backup** 或 **fault** 状态。

如果检查脚本返回非零，节点会进入 **backup** 状态，并且它拥有的任何 VIP 被重新分配。

notify 脚本

keepalived 将以下三个参数传递给 notify 脚本：

- **$1** - **group** 或 **instance**
- **$2** - **group** 或 **instance** 的名称
- **$3** - 新状态： **master**、**backup** 或 **fault**

先决条件

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

流程

1. 创建所需脚本，并创建 **ConfigMap** 对象来容纳它。脚本没有输入参数，并且必须返回 **0**（**OK**）和 **1**（**fail**）。
   检查脚本，***mycheckscript.sh***：

```
#!/bin/bash
   # Whatever tests are needed
   # E.g., send request and verify response
exit 0
```

2. 创建 **ConfigMap** 对象：

```
$ oc create configmap mycustomcheck --from-file=mycheckscript.sh
```

3. 将脚本添加到容器集。挂载的 **ConfigMap** 对象的 **defaultMode** 必须能够使用 **oc** 命令或编辑部署配置来运行。值通常为 **0755**、**493**（十进制）：

```
$ oc set env deploy/ipfailover-keepalived \
   OPENSHIFT_HA_CHECK_SCRIPT=/etc/keepalive/mycheckscript.sh
```

```
$ oc set volume deploy/ipfailover-keepalived --add --overwrite \
   --name=config-volume \
   --mount-path=/etc/keepalive \
   --source='{"configMap": { "name": "mycustomcheck", "defaultMode": 493}}'
```

注意

oc set env 命令对空格敏感。**=** 符号的两侧不能有空格。

提示

您还可以编辑 **ipfailover-keepalived** 部署配置：

```
$ oc edit deploy ipfailover-keepalived
```

```
    spec:
      containers:
      - env:
        - name: OPENSHIFT_HA_CHECK_SCRIPT (1)
          value: /etc/keepalive/mycheckscript.sh
...
        volumeMounts: (2)
        - mountPath: /etc/keepalive
          name: config-volume
      dnsPolicy: ClusterFirst
...
      volumes: (3)
      - configMap:
          defaultMode: 0755 (4)
          name: customrouter
        name: config-volume
...
```

1. 在 **spec.container.env** 字段中，添加 **OPENSHIFT_HA_CHECK_SCRIPT** 环境变量以指向挂载的脚本文件。
2. 添加 **spec.container.volumeMounts** 字段以创建挂载点。
3. 添加新的 **spec.volumes** 字段以提及配置映射。
4. 这将设置文件的运行权限。在重新读后，其显示为十进制 **493**。

保存更改并退出编辑器。这会重启 **ipfailover-keepalived**。

4.4. 配置 VRRP 抢占

当一个节点上的虚拟 IP（VIP）因为通过了检查脚本的检查而脱离 **fault** 状态时，如果其优先级低于当前处于 **master** 状态的节点上的 VIP，则节点上的 VIP 将进入 **backup** 状态。**nopreempt** 策略不会将 **master** 从主机上的较低优先级 VIP 移到主机上的优先级更高的 VIP。当使用默认的 **preempt_delay 300** 时，Keepalived 会等待指定的 300 秒，并将 **master** 移到主机上的优先级更高的 VIP。

流程

- 要指定抢占，输入 **oc edit deploy ipfailover-keepalived** 以编辑路由器部署配置：

```
$ oc edit deploy ipfailover-keepalived
```

```
...
    spec:
      containers:
      - env:
        - name: OPENSHIFT_HA_PREEMPTION (1)
          value: preempt_delay 300
...
```

1. 设置 **OPENSHIFT_HA_PREEMPTION** 值：
   - **preempt_delay 300**：Keepalived 会等待指定的 300 秒，并将 **master** 移到主机上的优先级更高的 VIP。这是默认值。
   - **nopreempt**：不会将 **master** 从主机上的较低优先级 VIP 移到主机上的优先级更高的 VIP。

4.5. 部署多个 IP 故障转移实例

每个 IP 转移 pod 由 IP 故障转移部署配置管理，每个节点 **1** 个 pod，以一个 Keepalived 守护进程运行。配置更多 IP 故障转移部署配置后，会创建更多 pod，更多的守护进程加入常见的虚拟路由器冗余协议（VRRP）协商。此协商由所有 Keepalived 守护进程完成，它决定了哪些节点服务是哪个虚拟 IP（VIP）。

Keepalived 内部为每个 VIP 分配一个唯一的 **vrrp-id**。协商使用这一组 **vrrp-ids**，在做出决策时，胜出的 **vrrp-id** 对应的 VIP 将在胜出的节点上服务。

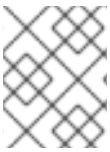
因此，对于 IP 故障转移部署配置中定义的每个 VIP，IP 故障转移 pod 必须分配对应的 **vrrp-id**。这可以从 **OPENSHIFT_HA_VRRP_ID_OFFSET** 开始，并按顺序将 **vrrp-ids** 分配到 VIP 列表来实现。**vrrp-ids** 的值可在 **1..255** 之间。

当存在多个 IP 故障转移部署配置时，您必须指定 **OPENSHIFT_HA_VRRP_ID_OFFSET**，以便在部署配置中增加 VIP 的数量，并且没有 **vrrp-id** 范围重叠。

4.6. 为超过 254 地址配置 IP 故障转移

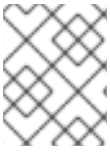
IP 故障转移管理有 254 个组虚拟 IP（VIP）地址的限制。默认情况下，OpenShift Container Platform 会为每个组分配一个 IP 地址。您可以使用 **OPENSHIFT_HA_VIP_GROUPS** 变量进行更改，使得每个组中有多个 IP 地址，并在配置 IP 故障转移时定义每个虚拟路由器冗余协议（VRRP）实例可用的 VIP 组数量。

在 VRRP 故障转移事件中，对 VIP 进行分组会为每个 VRRP 创建更广泛的 VIP 分配范围，并在集群中的所有主机都能够从本地访问服务时很有用。例如，当服务通过 **ExternalIP** 公开时。



注意

使用故障转移的一个规则是，请勿将路由等服务限制到一个特定的主机。相反，服务应复制到每一主机上，以便在 IP 故障转移时，不必在新主机上重新创建服务。



注意

如果使用 OpenShift Container Platform 健康检查，IP 故障转移和组的性质意味着不会检查组中的所有实例。因此，必须使用 Kubernetes 健康检查来确保服务处于活动状态。

先决条件

- 使用具有 **cluster-admin** 权限的用户登陆到集群。

流程

- 要更改分配给每个组的 IP 地址数量，请更改 **OPENSHIFT_HA_VIP_GROUPS** 变量的值，例如：

  IP 故障转换配置的 Deployment YAML 示例

```
...
  spec:
    env:
    - name: OPENSHIFT_HA_VIP_GROUPS 1
      value: "3"
...
```

1 如果在有七个 VIP 的环境中将 **OPENSHIFT_HA_VIP_GROUPS** 设置为 **3**，它会创建三个组，将三个 VIP 分配到第一个组，为剩余的两个组各分配两个 VIP。

注意

如果 **OPENSHIFT_HA_VIP_GROUPS** 设置的组数量少于设置为故障的 IP 地址数量，则组包含多个 IP 地址，且所有地址都作为一个单元移动。

4.7. EXTERNALIP 的高可用性

在非云集群中，可以组合使用 IP 故障切换和 **ExternalIP** 到服务。对于使用 **ExternalIP** 创建服务的用户，结果是高可用性服务。

方法是指定集群网络配置的 **spec.ExternalIP.autoAssignCIDRs** 范围，然后在创建 IP 故障转移配置时使用相同的范围。

因为 IP 故障转移最多可支持整个集群的 255 个 VIP，所以 **spec.ExternalIP.autoAssignCIDRs** 必须为 **/24** 或更小。

其他资源

- 配置 ExternalIP
- Kubernetes 文档 - ExternalIP

4.8. 删除 IP 故障切换

在初始配置 IP 故障切换时，集群中的 worker 节点会使用 **iptables** 规则修改，该规则明确允许 Keepalived 在 **224.0.0.18** 上多播数据包。由于对节点的更改，移除 IP 故障切换需要运行一个作业来删除 **iptables** 规则并删除 Keepalived 使用的虚拟 IP 地址。

流程

1. 可选：识别并删除存储为配置映射的任何检查和通知脚本：

   a. 确定任何用于 IP 故障切换的 pod 是否使用配置映射作为卷：

```
$ oc get pod -l ipfailover \
  -o jsonpath="\
{range .items[?(@.spec.volumes[*].configMap)]}
```

```
{'Namespace: '}{.metadata.namespace}
{'Pod:       '}{.metadata.name}
{'Volumes that use config maps:'}
{range .spec.volumes[?(@.configMap)]}  {'volume:    '}{.name}
  {'configMap: '}{.configMap.name}{'\n'}{end}
{end}"
```

输出示例

```
Namespace: default
Pod:       keepalived-worker-59df45db9c-2x9mn
Volumes that use config maps:
  volume:    config-volume
  configMap: mycustomcheck
```

b. 如果上一步提供了用作卷的配置映射的名称，请删除配置映射：

```
$ oc delete configmap <configmap_name>
```

2. 为 IP 故障切换识别现有部署：

```
$ oc get deployment -l ipfailover
```

输出示例

```
NAMESPACE   NAME        READY   UP-TO-DATE   AVAILABLE   AGE
default     ipfailover   2/2     2            2           105d
```

3. 删除部署：

```
$ oc delete deployment <ipfailover_deployment_name>
```

4. 删除 **ipfailover** 服务帐户：

```
$ oc delete sa ipfailover
```

5. 运行一个作业，该作业会删除最初配置 IP 故障切换时添加的 IP 表规则：

a. 创建一个文件，如 **remove-ipfailover-job.yaml**，其内容类似以下示例：

```
apiVersion: batch/v1
kind: Job
metadata:
  generateName: remove-ipfailover-
  labels:
    app: remove-ipfailover
spec:
  template:
    metadata:
      name: remove-ipfailover
    spec:
      containers:
      - name: remove-ipfailover
```

```
      image: registry.redhat.io/openshift4/ose-keepalived-ipfailover-rhel9:v4.19
      command: ["/var/lib/ipfailover/keepalived/remove-failover.sh"]
    nodeSelector: 1
      kubernetes.io/hostname: <host_name> 2
    restartPolicy: Never
```

1. **nodeSelector** 可能与旧 IP 故障转移部署中使用的选择器相同。
2. 为集群中配置 IP 故障切换的每个节点运行作业，并每次替换主机名。

b. 运行作业：

```
$ oc create -f remove-ipfailover-job.yaml
```

输出示例

```
job.batch/remove-ipfailover-2h8dm created
```

验证

- 确认作业删除了 IP 故障切换的初始配置。

```
$ oc logs job/remove-ipfailover-2h8dm
```

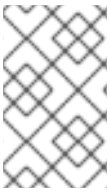
输出示例

```
remove-failover.sh: OpenShift IP Failover service terminating.
  - Removing ip_vs module ...
  - Cleaning up ...
  - Releasing VIPs  (interface eth0) ...
```

第 5 章 配置集群范围代理

生产环境可能会拒绝直接访问互联网，而是提供 HTTP 或 HTTPS 代理。您可以通过修改现有集群的 Proxy 对象或在新集群的 **install-config.yaml** 文件中配置代理设置，将 OpenShift Container Platform 配置为使用代理。

在支持的平台中为集群启用集群范围的出口代理后，Red Hat Enterprise Linux CoreOS (RHCOS)会使用支持的平台上存在的 **install-config.yaml** 文件中的 **networking.machineNetwork[].cidr**, **networking.clusterNetwork[].cidr**, 和 **networking.serviceNetwork[]** 字段的值填充 **status.noProxy** 参数。



注意

作为安装后任务，您可以更改 **networking.clusterNetwork[].cidr** 值，但不能更改 **networking.machineNetwork[].cidr** 和 **networking.serviceNetwork[]** 值。如需更多信息，请参阅"配置集群网络范围"。

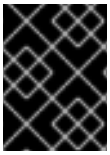
对于在 Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure 和 Red Hat OpenStack Platform (RHOSP) 上安装，**status.noProxy** 参数也会填充实例元数据端点 **169.254.169.254**。

由 RHCOS 添加到 Proxy 对象的 status: 段中的值示例

```
apiVersion: config.openshift.io/v1
kind: Proxy
metadata:
  name: cluster
# ...
networking:
  clusterNetwork: 1
  - cidr: <ip_address_from_cidr>
    hostPrefix: 23
  network type: OVNKubernetes
  machineNetwork: 2
  - cidr: <ip_address_from_cidr>
  serviceNetwork: 3
  - 172.30.0.0/16
# ...
status:
  noProxy:
  - localhost
  - .cluster.local
  - .svc
  - 127.0.0.1
  - <api_server_internal_url> 4
# ...
```

1. 指定从中分配 Pod IP 地址的 IP 地址块。默认值为 **10.128.0.0/14**，主机前缀为 **/23**。
2. 指定机器的 IP 地址块。默认值为 **10.0.0.0/16**。
3. 为服务指定 IP 地址块。默认值为 **172.30.0.0/16**。

4 您可以通过运行 **oc get infrastructures.config.openshift.io cluster -o jsonpath='{.status.etcdDiscoveryDomain}'** 命令来查找内部 API 服务器的 URL。



重要

如果您的安装类型不包括设置 **networking.machineNetwork[].cidr** 字段，则必须在 **.status.noProxy** 字段中手动包含机器 IP 地址，以确保节点之间的流量可以绕过代理。

5.1. 先决条件

查看集群需要访问的站点中的内容，决定这些站点中的任何站点是否需要绕过代理。默认情况下，所有集群系统的出站流量都需经过代理，包括对托管集群的云供应商 API 的调用。系统范围的代理仅影响系统组件，而不会影响用户工作负载。如有必要，将站点添加到 **Proxy** 对象的 **spec.noProxy** 参数，以绕过代理。

5.2. 启用集群范围代理

Proxy 对象用于管理集群范围出口代理。当在没有配置代理的情况下安装或升级集群时，仍会生成 **Proxy** 对象，但它有一个空的 **spec**。例如：

```
apiVersion: config.openshift.io/v1
kind: Proxy
metadata:
  name: cluster
spec:
  trustedCA:
    name: ""
status:
```

注意

只支持名为 **cluster** 的 **Proxy** 对象，且无法创建额外的代理。

集群管理员可以通过修改这个 **cluster Proxy** 对象来配置 OpenShift Container Platform 的代理。

警告

在为集群启用集群范围代理功能并保存 **Proxy** 对象文件后，Machine Config Operator (MCO) 会重启集群中的所有节点，以便每个节点可以访问集群外的连接。您不需要手动重新引导这些节点。

先决条件

- 集群管理员权限
- 已安装 OpenShift Container Platform **oc** CLI 工具

流程

1. 创建包含代理 HTTPS 连接所需的额外 CA 证书的 ConfigMap。

注意

如果代理的身份证书由 Red Hat Enterprise Linux CoreOS (RHCOS) 信任捆绑包的颁发机构签名，您可以跳过这一步。

a. 创建名为 **user-ca-bundle.yaml** 的文件，并提供 PEM 编码证书的值：

```
apiVersion: v1
data:
  ca-bundle.crt: | 1
    <MY_PEM_ENCODED_CERTS> 2
kind: ConfigMap
metadata:
  name: user-ca-bundle 3
  namespace: openshift-config 4
```

1. 这个数据键必须命名为 **ca-bundle.crt**。
2. 一个或多个 PEM 编码的 X.509 证书，用来为代理的身份证书签名。
3. 从 **Proxy** 对象引用的配置映射名称。
4. 配置映射必须存在于 **openshift-config** 命名空间中。

b. 输入以下命令从 **user-ca-bundle.yaml** 文件创建配置映射：

```
$ oc create -f user-ca-bundle.yaml
```

2. 使用 **oc edit** 命令修改 **Proxy** 对象：

```
$ oc edit proxy/cluster
```

3. 为代理配置所需的字段：

```
apiVersion: config.openshift.io/v1
kind: Proxy
metadata:
  name: cluster
spec:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
  readinessEndpoints:
  - http://www.google.com 4
  - https://www.google.com
  trustedCA:
    name: user-ca-bundle 5
```

1. 用于创建集群外 HTTP 连接的代理 URL。URL 方案必须是 **http**。

2. 用于创建集群外 HTTPS 连接的代理 URL。URL 方案必须是 **http** 或 **https**。指定支持 URL 方案的代理的 URL。例如，如果大多数代理被配置为使用 **https**，则大多数代理都会报告错误，但它们只支持 **http**。此失败消息可能无法传播到日志，并可能显示为网络连接失败。如果使用侦听来自集群的 **https** 连接的代理，您可能需要配置集群以接受代理使用的 CA 和证书。

3. 要排除代理的目标域名、域、IP 地址（或其他网络 CIDR）和端口号的列表（以逗号分隔）。

   注意

   只有在配置 IPv6 地址时，才支持端口号。配置 IPv4 地址时不支持端口号。

   在域前面加 **.** 来仅匹配子域。例如： **.y.com** 匹配 **x.y.com**，但不匹配 **y.com**。使用 * 可对所有目的地绕过所有代理。

   如果您的 **noproxy** 字段需要包含域地址，您必须在 **noproxy** 字段中明确指定该 FQDN 或前缀匹配子域。您不能使用封装域的 IP 地址或 CIDR 范围。这是因为集群不会在分配路由连接前等待 DNS 返回 IP 地址，并根据请求明确检查。例如，如果您有一个 CIDR 块值，如 **10.0.0.0/24**，**noproxy** 字段和字段尝试访问 **https://10.0.0.11**，则地址可以成功匹配。但是，尝试访问 **https://exampleserver.externaldomain.com**（其 A 记录条目为 **10.0.0.11**）会失败。对于 **noproxy** 字段，还需要额外的 **.externaldomain.com** 值。

   如果您扩展了未包含在安装配置中 **networking.machineNetwork[].cidr** 字段定义的计算节点，您必须将它们添加到此列表中，以防止连接问题。

   如果未设置 **httpProxy** 或 **httpsProxy** 字段，则此字段将被忽略。

4. 将 **httpProxy** 和 **httpsProxy** 值写进状态之前，执行就绪度检查时要使用的一个或多个集群外部 URL。

5. 引用 **openshift-config** 命名空间中的 ConfigMap，其包含代理 HTTPS 连接所需的额外 CA 证书。注意 ConfigMap 必须已经存在，然后才能在这里引用它。此字段是必需的，除非代理的身份证书由来自 RHCOS 信任捆绑包的颁发机构签名。

4. 保存文件以应用更改。

5.3. 删除集群范围代理服务器

cluster Proxy 对象不能被删除。要从集群中删除代理，请删除 Proxy 对象的所有 **spec** 字段。

先决条件

- 集群管理员权限
- 已安装 OpenShift Container Platform **oc** CLI 工具

流程

1. 使用 **oc edit** 命令来修改代理：

```
$ oc edit proxy/cluster
```

2. 删除 Proxy 对象的所有 **spec** 字段。例如：

```
apiVersion: config.openshift.io/v1
kind: Proxy
metadata:
  name: cluster
spec: {}
```

3. 保存文件以使改变生效。

5.4. 验证集群范围代理配置

部署集群范围代理配置后，您可以验证它是否按预期工作。按照以下步骤检查日志并验证实施。

先决条件

- 有集群管理员权限。
- 已安装 OpenShift Container Platform **oc** CLI 工具。

流程

1. 使用 **oc** 命令检查代理配置状态：

```
$ oc get proxy/cluster -o yaml
```

2. 验证输出中的代理字段，以确保它们与您的配置匹配。具体来说，检查 **spec.httpProxy**,**spec.httpsProxy**,**spec.noProxy**, 和 **spec.trustedCA** 字段。

3. 检查 **Proxy** 对象的状态：

```
$ oc get proxy/cluster -o jsonpath='{.status}'
```

输出示例

```
{
status:
    httpProxy: http://user:xxx@xxxx:3128
    httpsProxy: http://user:xxx@xxxx:3128
    noProxy:
.cluster.local,.svc,10.0.0.0/16,10.128.0.0/14,127.0.0.1,169.254.169.254,172.30.0.0/16,localhost,
test.no-proxy.com
}
```

4. 检查 Machine Config Operator (MCO) 的日志，以确保成功应用配置更改：

```
$ oc logs -n openshift-machine-config-operator $(oc get pods -n openshift-machine-config-
operator -l k8s-app=machine-config-operator -o name)
```

5. 查找指示代理设置的消息，并在需要时重启节点。

6. 通过检查发出外部请求的组件的日志来验证系统组件是否使用代理，如 Cluster Version Operator (CVO)：

```
$ oc logs -n openshift-cluster-version $(oc get pods -n openshift-cluster-version -l k8s-app=machine-config-operator -o name)
```

7. 查找显示外部请求已通过代理路由的日志条目。

其他资源

- 配置集群网络范围
- 了解 CA 捆绑包证书
- 代理证书
- 集群范围的代理设置如何应用到 OpenShift Container Platform 节点？

第 6 章 配置自定义 PKI

有些平台组件，如 Web 控制台，使用 Routes 进行通信，且必须信任其他组件的证书与其交互。如果您使用的是自定义公钥基础架构 (PKI)，您必须将其配置为在集群中识别其私有签名的 CA 证书。

您可以使用 Proxy API 添加集群范围的可信 CA 证书。您必须在安装过程中或运行时执行此操作。

- 在 *安装过程* 中，配置集群范围的代理。您需要在 **install-config.yaml** 文件中的 **additionalTrustBundle** 设置中定义私有签名的 CA 证书。
  安装程序生成名为 **user-ca-bundle** 的 ConfigMap，其中包含您定义的附加 CA 证书。然后，Cluster Network Operator 会创建 **trusted-ca-bundle** ConfigMap，将这些 CA 证书与 Red Hat Enterprise Linux CoreOS (RHCOS)信任捆绑包合并；Proxy 对象的 **trustedCA** 字段中会引用此 ConfigMap。

- 在*运行时*，，修改默认 Proxy 对象使其包含您私有签名的 CA 证书 （集群代理启用工作流程的一部分）。这涉及创建包含集群应信任的私有签名 CA 证书的 ConfigMap，然后使用 **trustedCA** 引用私有签名证书的 ConfigMap 修改代理服务器资源。

注意

安装程序配置的 **additionalTrustBundle** 字段和 proxy 资源的 **trustedCA** 字段被用来管理集群范围信任捆绑包; 在安装时会使用 **additionalTrustBundle**，并在运行时使用代理的**trustedCA**。

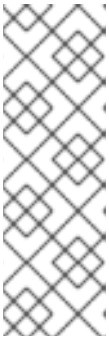
trustedCA 字段是对包含集群组件使用的自定义证书和密钥对的 **ConfigMap** 的引用。

6.1. 在安装过程中配置集群范围的代理

生产环境可能会拒绝直接访问互联网，而是提供 HTTP 或 HTTPS 代理。您可以通过在 **install-config.yaml** 文件中配置代理设置，将新的 OpenShift Container Platform 集群配置为使用代理。

先决条件

- 您有一个现有的 **install-config.yaml** 文件。
- 您检查了集群需要访问的站点，并确定它们中的任何站点是否需要绕过代理。默认情况下，所有集群出口流量都经过代理，包括对托管云供应商 API 的调用。如果需要，您将在 **Proxy** 对象的 **spec.noProxy** 字段中添加站点来绕过代理。



注意

Proxy 对象 **status.noProxy** 字段使用安装配置中的 **networking.machineNetwork[].cidr**、**networking.clusterNetwork[].cidr** 和 **networking.serviceNetwork[]** 字段的值填充。

对于在 Amazon Web Services(AWS)、Google Cloud Platform(GCP)、Microsoft Azure 和 Red Hat OpenStack Platform(RHOSP)上安装，**Proxy** 对象 **status.noProxy** 字段也会使用实例元数据端点填充(**169.254.169.254**)。

流程

1. 编辑 **install-config.yaml** 文件并添加代理设置。例如：

```
apiVersion: v1
baseDomain: my.domain.com
proxy:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
additionalTrustBundle: | 4
    -----BEGIN CERTIFICATE-----
    <MY_TRUSTED_CA_CERT>
    -----END CERTIFICATE-----
additionalTrustBundlePolicy: <policy_to_add_additionalTrustBundle> 5
```

1. 用于创建集群外 HTTP 连接的代理 URL。URL 方案必须是 **http**。
2. 用于创建集群外 HTTPS 连接的代理 URL。
3. 要从代理中排除的目标域名、IP 地址或其他网络 CIDR 的逗号分隔列表。在域前面加上 **.** 以仅匹配子域。例如，**.y.com** 匹配 **x.y.com**，但不匹配 **y.com**。使用 * 绕过所有目的地的代理。
4. 如果提供，安装程序会在 **openshift-config** 命名空间中生成名为 **user-ca-bundle** 的配置映射，其包含代理 HTTPS 连接所需的一个或多个额外 CA 证书。然后，Cluster Network Operator 会创建一个 **trusted-ca-bundle** 配置映射，将这些内容与 Red Hat Enterprise Linux CoreOS (RHCOS)信任捆绑包合并，**Proxy** 对象的 **trustedCA** 字段中也会引用此配置映射。**additionalTrustBundle** 字段是必需的，除非代理的身份证书由来自 RHCOS 信任捆绑包的颁发机构签名。
5. 可选：决定 **Proxy** 对象的配置以引用 **trustedCA** 字段中 **user-ca-bundle** 配置映射的策略。允许的值是 **Proxyonly** 和 **Always**。仅在配置了 **http/https** 代理时，使用 **Proxyonly** 引用 **user-ca-bundle** 配置映射。使用 **Always** 始终引用 **user-ca-bundle** 配置映射。默认值为 **Proxyonly**。

注意

安装程序不支持代理的 **readinessEndpoints** 字段。

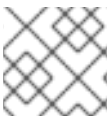
注意

如果安装程序超时，重启并使用安装程序的 **wait-for** 命令完成部署。例如：

```
$ ./openshift-install wait-for install-complete --log-level debug
```

2. 保存该文件并在安装 OpenShift Container Platform 时引用。

安装程序会创建一个名为 cluster 的集群范围代理，该代理 **使用** 提供的 **install-config.yaml** 文件中的代理设置。如果没有提供代理设置，仍然会创建一个 **cluster Proxy** 对象，但它会有一个空 **spec**。



注意

只支持名为 **cluster** 的 **Proxy** 对象，且无法创建额外的代理。

6.2. 启用集群范围代理

Proxy 对象用于管理集群范围出口代理。当在没有配置代理的情况下安装或升级集群时，仍会生成 **Proxy** 对象，但它有一个空的 **spec**。例如：

```
apiVersion: config.openshift.io/v1
kind: Proxy
metadata:
  name: cluster
spec:
  trustedCA:
    name: ""
status:
```

注意

只支持名为 **cluster** 的 **Proxy** 对象，且无法创建额外的代理。

集群管理员可以通过修改这个 **cluster Proxy** 对象来配置 OpenShift Container Platform 的代理。

警告

在为集群启用集群范围代理功能并保存 **Proxy** 对象文件后，Machine Config Operator (MCO) 会重启集群中的所有节点，以便每个节点可以访问集群外的连接。您不需要手动重新引导这些节点。

先决条件

- 集群管理员权限
- 已安装 OpenShift Container Platform **oc** CLI 工具

流程

1. 创建包含代理 HTTPS 连接所需的额外 CA 证书的 ConfigMap。

注意

如果代理的身份证书由 Red Hat Enterprise Linux CoreOS (RHCOS)信任捆绑包的颁发机构签名，您可以跳过这一步。

a. 创建名为 **user-ca-bundle.yaml** 的文件，并提供 PEM 编码证书的值：

```
apiVersion: v1
data:
  ca-bundle.crt: | 1
    <MY_PEM_ENCODED_CERTS> 2
kind: ConfigMap
```

```
  metadata:
    name: user-ca-bundle 3
    namespace: openshift-config 4
```

1 这个数据键必须命名为 **ca-bundle.crt**。

2 一个或多个 PEM 编码的 X.509 证书，用来为代理的身份证书签名。

3 从 **Proxy** 对象引用的配置映射名称。

4 配置映射必须存在于 **openshift-config** 命名空间中。

b. 输入以下命令从 **user-ca-bundle.yaml** 文件创建配置映射：

```
$ oc create -f user-ca-bundle.yaml
```

2. 使用 **oc edit** 命令修改 **Proxy** 对象：

```
$ oc edit proxy/cluster
```

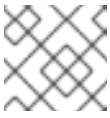
3. 为代理配置所需的字段：

```
apiVersion: config.openshift.io/v1
kind: Proxy
metadata:
  name: cluster
spec:
  httpProxy: http://<username>:<pswd>@<ip>:<port> 1
  httpsProxy: https://<username>:<pswd>@<ip>:<port> 2
  noProxy: example.com 3
  readinessEndpoints:
  - http://www.google.com 4
  - https://www.google.com
  trustedCA:
    name: user-ca-bundle 5
```

1 用于创建集群外 HTTP 连接的代理 URL。URL 方案必须是 **http**。

2 用于创建集群外 HTTPS 连接的代理 URL。URL 方案必须是 **http** 或 **https**。指定支持 URL 方案的代理的 URL。例如，如果大多数代理被配置为使用 **https**，则大多数代理都会报告错误，但它们只支持 **http**。此失败消息可能无法传播到日志，并可能显示为网络连接失败。如果使用侦听来自集群的 **https** 连接的代理，您可能需要配置集群以接受代理使用的 CA 和证书。

3 要排除代理的目标域名、域、IP 地址（或其他网络 CIDR）和端口号的列表（以逗号分隔）。



注意

只有在配置 IPv6 地址时，才支持端口号。配置 IPv4 地址时不支持端口号。

在域前面加 **.** 来仅匹配子域。例如： **.y.com** 匹配 **x.y.com**，但不匹配 **y.com**。使用 ***** 可对所有目的地绕过所有代理。

如果您的 **noproxy** 字段需要包含域地址，您必须在 **noproxy** 字段中明确指定该 FQDN 或前缀匹配子域。您不能使用封装域的 IP 地址或 CIDR 范围。这是因为集群不会在分配路由连接前等待 DNS 返回 IP 地址，并根据请求明确检查。例如，如果您有一个 CIDR 块值，如 **10.0.0.0/24**，**noproxy** 字段和字段尝试访问 **https://10.0.0.11**，则地址可以成功匹配。但是，尝试访问 **https://exampleserver.externaldomain.com**（其 A 记录条目为 **10.0.0.11**）会失败。对于 **noproxy** 字段，还需要额外的 **.externaldomain.com** 值。

如果您扩展了未包含在安装配置中 **networking.machineNetwork[].cidr** 字段定义的计算节点，您必须将它们添加到此列表中，以防止连接问题。

如果未设置 **httpProxy** 或 **httpsProxy** 字段，则此字段将被忽略。

4 将 **httpProxy** 和 **httpsProxy** 值写进状态之前，执行就绪度检查时要使用的一个或多个集群外部 URL。

5 引用 **openshift-config** 命名空间中的 ConfigMap，其包含代理 HTTPS 连接所需的额外 CA 证书。注意 ConfigMap 必须已经存在，然后才能在这里引用它。此字段是必需的，除非代理的身份证书由来自 RHCOS 信任捆绑包的颁发机构签名。

4. 保存文件以使改变生效。

6.3. 使用 OPERATOR 进行证书注入

在您的自定义 CA 证书通过 ConfigMap 添加到集群中后，Cluster Network Operator 会将用户提供的证书和系统 CA 证书合并到单一捆绑包中，并将合并的捆绑包注入请求信任捆绑包注入的 Operator。

重要

在配置映射中添加 **config.openshift.io/inject-trusted-cabundle="true"** 标签后，会删除其中的现有数据。Cluster Network Operator 获取配置映射的所有权，并只接受 **ca-bundle** 作为数据。您必须使用单独的配置映射存储 **service-ca.crt**，方法是使用 **service.beta.openshift.io/inject-cabundle=true** 注解或类似的配置。在同一配置映射中添加 **config.openshift.io/inject-trusted-cabundle="true"** 标签和 **service.beta.openshift.io/inject-cabundle=true** 注解可能会导致问题。

Operator 通过创建一个带有以下标签的空 ConfigMap 来请求此注入：

```
config.openshift.io/inject-trusted-cabundle="true"
```

空 ConfigMap 示例：

```
apiVersion: v1
data: {}
kind: ConfigMap
metadata:
  labels:
    config.openshift.io/inject-trusted-cabundle: "true"
  name: ca-inject 1
  namespace: apache
```

1 指定空 ConfigMap 名称。

Operator 将这个 ConfigMap 挂载到容器的本地信任存储中。

注意

只有在 Red Hat Enterprise Linux CoreOS (RHCOS)信任捆绑包中没有包括证书时，才需要添加可信 CA 证书。

证书注入不仅限于 Operator。当使用 **config.openshift.io/inject-trusted-cabundle=true**标记（label) 创建一个空的 ConfigMap 时，Cluster Network Operator会跨命名空间注入证书 。

ConfigMap 可以驻留在任何命名空间中，但 ConfigMap 必须作为卷挂载到需要自定义 CA 的 Pod 中的每个容器。例如：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-example-custom-ca-deployment
  namespace: my-example-custom-ca-ns
spec:
  ...
    spec:
      ...
      containers:
        - name: my-container-that-needs-custom-ca
          volumeMounts:
          - name: trusted-ca
            mountPath: /etc/pki/ca-trust/extracted/pem
            readOnly: true
      volumes:
      - name: trusted-ca
        configMap:
          name: ca-inject
          items:
            - key: ca-bundle.crt 1
              path: tls-ca-bundle.pem 2
```

1 **CA-bundle.crt** 需要作为 ConfigMap 密钥。

2 **TLS-ca-bundle.pem** 需要作为 ConfigMap 的路径。