



Red Hat

Red Hat Enterprise Linux 8

配置和管理网络

管理网络接口、防火墙和高级网络功能

管理网络接口、防火墙和高级网络功能

Legal Notice

Copyright © 2025 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

Java® is a registered trademark of Oracle and/or its affiliates.

XFS® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

使用 Red Hat Enterprise Linux (RHEL) 的网络功能，您可以配置主机以满足您组织的网络和安全要求。例如：您可以配置绑定、VLAN、网桥、隧道和其他网络类型，来将主机连接到网络。您可以为本地主机和整个网络构建性能关键防火墙。RHEL 包含数据包过滤软件，如 firewalld 服务、nftables 框架和 Express Data Path(XDP)。RHEL 还支持高级网络功能，如基于策略的路由和多路径 TCP (MPTCP)。

Table of Contents

对红帽文档提供反馈	9
第 1 章 实现一致的网络接口命名	10
1.1. UDEV 设备管理器如何重命名网络接口	10
1.2. 网络接口命名策略	11
1.3. 网络接口命名方案	11
1.4. 切换到不同的网络接口命名方案	12
1.5. 在 IBM Z 平台上确定可预测的 ROCE 设备名称	13
1.6. 在安装过程中为以太网接口自定义前缀	15
1.7. 使用 UDEV 规则配置用户定义的网络接口名称	16
1.8. 使用 SYSTEMD 链接文件配置用户定义的网络接口名称	18
1.9. 使用 SYSTEMD 链接文件将替代名称分配给网络接口	20
第 2 章 配置以太网连接	22
2.1. 使用 NMCLI 配置以太网连接	22
2.2. 使用 NMCLI 交互式编辑器配置以太网连接	25
2.3. 使用 NMTUI 配置以太网连接	27
2.4. 使用控制中心配置以太网连接	31
2.5. 使用 NM-CONNECTION-EDITOR 配置以太网连接	32
2.6. 使用带有接口名称的 NMSTATECTL 使用静态 IP 地址配置以太网连接	35
2.7. 使用 NETWORK RHEL 系统角色和接口名称，配置具有静态 IP 地址的以太网连接	37
2.8. 使用 NETWORK RHEL 系统角色和设备路径，配置具有静态 IP 地址的以太网连接	39
2.9. 使用带有接口名称的 NMSTATECTL 使用动态 IP 地址配置以太网连接	41
2.10. 使用 NETWORK RHEL 系统角色和接口名称，配置具有动态 IP 地址的以太网连接	43
2.11. 使用 NETWORK RHEL 系统角色和设备路径，配置具有动态 IP 地址的以太网连接	45
2.12. 按接口名称，使用单个连接配置文件配置多个以太网接口	48
2.13. 使用 PCI ID 为多个以太网接口配置一个连接配置文件	48
第 3 章 配置网络绑定	50
3.1. 了解控制器和端口接口的默认行为	50
3.2. 依赖绑定模式的上游交换机配置	50
3.3. 使用 NMCLI 配置网络绑定	51
3.4. 使用 RHEL WEB 控制台配置网络绑定	53
3.5. 使用 NMTUI 配置网络绑定	56
3.6. 使用 NM-CONNECTION-EDITOR 配置网络绑定	60
3.7. 使用 NMSTATECTL 配置网络绑定	62
3.8. 使用 NETWORK RHEL 系统角色配置网络绑定	64
3.9. 创建网络绑定以便在不中断 VPN 的情况下在以太网和无线连接间进行切换	66
3.10. 不同的网络绑定模式	69
3.11. XMIT_HASH_POLICY BONDING 参数	70
第 4 章 配置 NIC TEAM	72
4.1. 了解控制器和端口接口的默认行为	72
4.2. 了解 TEAMD 服务、运行程序和 LINK-WATCHERS	72
4.3. 使用 NMCLI 配置 NIC TEAM	73
4.4. 使用 RHEL WEB 控制台配置 NIC TEAM	76
4.5. 使用 NM-CONNECTION-EDITOR 配置 NIC TEAM	79
第 5 章 配置 VLAN 标记	82
5.1. 使用 NMCLI 配置 VLAN 标记	82
5.2. 使用 RHEL WEB 控制台配置 VLAN 标记	84
5.3. 使用 NMTUI 配置 VLAN 标记	86

5.4. 使用 NM-CONNECTION-EDITOR 配置 VLAN 标记	90
5.5. 使用 NMSTATECTL 配置 VLAN 标记	92
5.6. 使用 NETWORK RHEL 系统角色配置 VLAN 标记	94
第 6 章 配置网络桥接	97
6.1. 使用 NMCLI 配置网桥	97
6.2. 使用 RHEL WEB 控制台配置网桥	100
6.3. 使用 NMTUI 配置网桥	102
6.4. 使用 NM-CONNECTION-EDITOR 配置网桥	106
6.5. 使用 NMSTATECTL 配置网桥	108
6.6. 使用 NETWORK RHEL 系统角色配置网桥	110
第 7 章 设置 IPSEC VPN	113
7.1. LIBRESWAN 作为 IPSEC VPN 的实现	113
7.2. LIBRESWAN 中的身份验证方法	114
7.3. 安装 LIBRESWAN	115
7.4. 创建主机到主机的 VPN	116
7.5. 配置站点到站点的 VPN	117
7.6. 配置远程访问 VPN	118
7.7. 配置网格 VPN	119
7.8. 部署符合 FIPS 的 IPSEC VPN	122
7.9. 使用密码保护 IPSEC NSS 数据库	124
7.10. 配置 IPSEC VPN 以使用 TCP	126
7.11. 配置自动检测和使用 ESP 硬件卸载来加速 IPSEC 连接	126
7.12. 在绑定中配置 ESP 硬件卸载以加快 IPSEC 连接	127
7.13. 使用 RHEL 系统角色配置 IPSEC VPN 连接	129
7.14. 配置选择不使用系统范围的加密策略的 IPSEC 连接	136
7.15. IPSEC VPN 配置故障排除	136
7.16. 使用 CONTROL-CENTER 配置 VPN 连接	140
7.17. 使用 NM-CONNECTION-EDITOR 配置 VPN 连接	146
7.18. 将 VPN 连接分配给专用路由表，以防止连接绕过隧道	151
7.19. 其他资源	153
第 8 章 配置 IP 隧道	154
8.1. 配置 IPIP 隧道来封装 IPV4 数据包中的 IPV4 流量	155
8.2. 配置 GRE 隧道来封装 IPV4 数据包中的第 3 层流量	158
8.3. 配置 GRETAP 隧道来通过 IPV4 传输以太网帧	161
第 9 章 使用 VXLAN 为虚拟机创建虚拟第 2 层域	166
9.1. VXLAN 的优点	166
9.2. 在主机上配置以太网接口	167
9.3. 创建附加了 VXLAN 的网桥	168
9.4. 在带有现有网桥的 LIBVIRT 中创建一个虚拟网络	170
9.5. 配置虚拟机以使用 VXLAN	172
第 10 章 管理 WIFI 连接	175
10.1. 支持的 WIFI 安全类型	175
10.2. 使用 NMCLI 连接到 WIFI 网络	176
10.3. 使用 GNOME 系统菜单连接到 WIFI 网络	179
10.4. 使用 GNOME 设置应用程序连接到 WIFI 网络	181
10.5. 使用 NMTUI 配置 WIFI 连接	182
10.6. 使用 NM-CONNECTION-EDITOR 配置 WIFI 连接	185
10.7. 使用 NETWORK RHEL 系统角色配置带有 802.1X 网络身份验证的 WIFI 连接	187
10.8. 使用 NMCLI 在现有配置文件中配置带有 802.1X 网络身份验证的 WIFI 连接	190

10.9. 手动设置无线规范域	192
第 11 章 将 RHEL 配置为 WPA2 或 WPA3 个人访问令牌	195
第 12 章 使用 MACSEC 加密同一物理网络中的第 2 层流量	199
12.1. MACSEC 如何提高安全性	199
12.2. 使用 NMCLI 配置 MACSEC 连接	200
第 13 章 开始使用 IPVLAN	203
13.1. IPVLAN 模式	203
13.2. IPVLAN 和 MACVLAN 的比较	203
13.3. 使用 IPROUTE2 创建和配置 IPVLAN 设备	204
第 14 章 配置 NETWORKMANAGER 以忽略某些设备	206
14.1. 永久将设备配置为网络管理器（NETWORKMANAGER）中非受管设备	206
14.2. 将设备临时配置为在 NETWORKMANAGER 中不被管理	208
第 15 章 使用 NMCLI 配置回环接口	210
第 16 章 创建一个虚拟接口	212
16.1. 使用 NMCLI 创建一个同时具有 IPV4 和 IPV6 地址的虚拟接口	212
第 17 章 使用 NETWORKMANAGER 为特定连接禁用 IPV6	213
17.1. 使用 NMCLI 禁用连接上的 IPV6	213
第 18 章 更改主机名	215
18.1. 使用 NMCLI 更改主机名	215
18.2. 使用 HOSTNAMECTL 更改主机名	216
第 19 章 配置 NETWORKMANAGER DHCP 设置	218
19.1. 更改 NETWORKMANAGER 的 DHCP 客户端	218
19.2. 配置 NETWORKMANAGER 连接的 DHCP 超时行为	219
第 20 章 使用 NETWORKMANAGER 的一个分配程序脚本运行 HCLIENT EXIT HOOKS	221
20.1. NETWORKMANAGER 分配程序脚本的概念	221
20.2. 创建运行 DHCLIENT EXIT HOOKS 的 NETWORKMANAGER 分配程序脚本	222
第 21 章 手动配置 /ETC/RESOLV.CONF 文件	224
21.1. 在 NETWORKMANAGER 配置中禁用 DNS 处理	224
21.2. 使用符号链接替换 /ETC/RESOLV.CONF 来手动配置 DNS 设置	225
第 22 章 配置 DNS 服务器顺序	227
22.1. NETWORKMANAGER 如何在 /ETC/RESOLV.CONF 中对 DNS 服务器进行排序	227
22.2. 设置 NETWORKMANAGER 范围默认 DNS 服务器优先级值	229
22.3. 设置网络管理器连接的 DNS 优先级	230
第 23 章 在不同域中使用不同的 DNS 服务器	232
23.1. 在 NETWORKMANAGER 中使用 DNSMASQ 将特定域的 DNS 请求发送到所选的 DNS 服务器	232
23.2. 使用 NETWORKMANAGER 中的 SYSTEMD-RESOLVED 将对特定域的 DNS 请求发送到所选的 DNS 服务器	235
第 24 章 管理默认网关设置	239
24.1. 使用 NMCLI 对现有连接设置默认网关	239
24.2. 使用 NMCLI 交互模式对现有连接设置默认网关	240
24.3. 使用 NM-CONNECTION-EDITOR 在现有连接上设置默认网关	242
24.4. 使用 CONTROL-CENTER 在现有连接上设置默认网关	244
24.5. 使用 NMSTATECTL 对现有连接设置默认网关	246

24.6. 使用 NETWORK RHEL 系统角色对现有连接设置默认网关	247
24.7. 使用旧的网络脚本在现有连接中设置默认网关	250
24.8. NETWORKMANAGER 如何管理多个默认网关	251
24.9. 配置 NETWORKMANAGER 以避免使用特定配置集提供默认网关	252
24.10. 修复因为多个默认网关导致的意外路由行为	253
第 25 章 配置静态路由	256
25.1. 需要静态路由的网络示例	256
25.2. 如何使用 NMCLI 工具配置静态路由	259
25.3. 使用 NMCLI 配置静态路由	261
25.4. 使用 NMTUI 配置静态路由	262
25.5. 使用 CONTROL-CENTER 配置静态路由	265
25.6. 使用 NM-CONNECTION-EDITOR 配置静态路由	267
25.7. 使用 NMCLI 交互模式配置静态路由	270
25.8. 使用 NMSTATECTL 配置静态路由	272
25.9. 使用 NETWORK RHEL 系统角色配置静态路由	273
25.10. 使用旧的网络脚本以键-值格式创建静态路由配置文件	276
25.11. 在使用旧的网络脚本时，使用 IP-COMMAND-FORMAT 创建静态路由配置文件	278
第 26 章 配置基于策略的路由以定义其他路由	281
26.1. 使用 NMCLI 将流量从特定子网路由到不同的默认网关	281
26.2. 使用 NETWORK RHEL 系统角色将流量从特定子网路由到不同的默认网关	286
26.3. 使用旧网络脚本时，涉及基于策略的路由的配置文件概述	292
26.4. 使用旧的网络脚本将来自特定子网的流量路由到不同的默认网关	293
第 27 章 在不同的接口上重复使用相同的 IP 地址	301
27.1. 在不同接口上永久重复使用相同的 IP 地址	301
27.2. 在不同接口中临时重复使用相同的 IP 地址	303
27.3. 其他资源	305
第 28 章 在隔离的 VRF 网络内启动服务	306
28.1. 配置 VRF 设备	306
28.2. 在隔离的 VRF 网络内启动服务	308
第 29 章 在 NETWORKMANAGER 连接配置文件中配置 ETHTOOL 设置	311
29.1. 使用 NMCLI 配置 ETHTOOL 卸载功能	311
29.2. 使用 NETWORK RHEL 系统角色配置 ETHTOOL 卸载功能	312
29.3. 使用 NMCLI 配置 ETHTOOL COALESCE 设置	314
29.4. 使用 NETWORK RHEL 系统角色配置 ETHTOOL COALESCE 设置	315
29.5. 使用 NMCLI 增加环缓冲区的大小，以减少数据包丢弃率	318
29.6. 使用 NETWORK RHEL 系统角色增加环缓冲区的大小，以减少高数据包丢弃率	320
第 30 章 网络管理器调试介绍	324
30.1. NETWORKMANAGER 重新应用方法的简介	324
30.2. 设置 NETWORKMANAGER 日志级别	329
30.3. 使用 NMCLI 临时设置运行时的日志级别	330
30.4. 查看 NETWORKMANAGER 日志	331
30.5. 调试级别和域	332
第 31 章 使用 LLDP 来调试网络配置问题	333
31.1. 使用 LLDP 信息调试不正确的 VLAN 配置	333
第 32 章 LINUX 流量控制	336
32.1. 排队规则概述	336
32.2. 使用 TC 工具检查网络接口的 QDISCS	337

32.3. 更新默认的 QDISC	337
32.4. 使用 TC 工具临时设置网络接口的当前 QDISC	339
32.5. 使用 NETWORKMANAGER 永久设置当前网络接口的 QDISC	339
32.6. RHEL 中可用的 QDISCS	340
第 33 章 使用 802.1X 标准和存储在文件系统上的证书对 RHEL 客户端进行网络验证	343
33.1. 使用 NMCLI 在现有以太网连接上配置 802.1X 网络身份验证	343
33.2. 使用 NMSTATECTL 配置带有 802.1X 网络身份验证的静态以太网连接	345
33.3. 使用 NETWORK RHEL 系统角色配置具有 802.1X 网络身份验证的静态以太网连接	347
第 34 章 使用带有 FREERADIUS 后端的 HOSTAPD 为 LAN 客户端设置 802.1X 网络身份验证服务	352
34.1. 先决条件	352
34.2. 在验证器中设置网桥	352
34.3. 配置 FREERADIUS，以使用 EAP 安全地验证网络客户端	354
34.4. 在有线网络中将 HOSTAPD 配置为验证器	360
34.5. 针对 FREERADIUS 服务器或验证器测试 EAP-TTLS 身份验证	363
34.6. 根据 HOSTAPD 身份验证事件阻止和允许流量	365
第 35 章 多路径 TCP 入门	369
35.1. 了解 MPTCP	369
35.2. 准备 RHEL 启用 MPTCP 支持	370
35.3. 使用 IPROUTE2 为 MPTCP 应用程序临时配置和启用多个路径	373
35.4. 为 MPTCP 应用程序永久配置多路径	376
35.5. 监控 MPTCP 子流	379
35.6. 在内核中禁用多路径 TCP	383
第 36 章 RHEL 中支持旧的网络脚本	384
36.1. 安装旧的网络脚本	384
第 37 章 使用 IFCFG 文件配置 IP 网络	385
37.1. 使用 IFCFG 文件配置带有静态网络设置的接口	385
37.2. 使用 IFCFG 文件配置带有动态网络设置的接口	386
37.3. 使用 IFCFG 文件管理系统范围以及专用连接配置集	387
第 38 章 KEYFILE 格式的 NETWORKMANAGER 连接配置文件	388
38.1. NETWORKMANAGER 配置文件的密钥文件格式	388
38.2. 使用 NMCLI 在离线模式下创建 KEYFILE 连接配置文件	389
38.3. 手动创建 KEYFILE 格式的 NETWORKMANAGER 配置文件	391
38.4. IFCFG 和 KEYFILE 格式的配置文件在接口重命名方面的差异	393
38.5. 将 NETWORKMANAGER 配置集从 IFCFG 迁移到 KEYFILE 格式	394
第 39 章 SYSTEMD 网络目标和服务	396
39.1. 网络和网络在线 SYSTEMD 目标的不同	396
39.2. NETWORKMANAGER-WAIT-ONLINE 概述	396
39.3. 将 SYSTEMD 服务配置为在网络已启动后再启动	397
第 40 章 NMSTATE 简介	399
40.1. 在 PYTHON 应用程序中使用 LIBNMSTATE 库	399
40.2. 使用 NMSTATECTL 更新当前网络配置	400
40.3. NETWORK RHEL 系统角色的网络状态	401
第 41 章 使用和配置 FIREWALLD	404
41.1. 使用 FIREWALLD、NFTABLES 或者 IPTABLES 时	404
41.2. 防火墙区域	405
41.3. 防火墙策略	408
41.4. 防火墙规则	409

41.5. 防火墙直接规则	410
41.6. 预定义的 FIREWALLD 服务	410
41.7. 使用 FIREWALLD 区	411
41.8. 使用 FIREWALLD 控制网络流量	425
41.9. 在区域间过滤转发的流量	433
41.10. 使用 FIREWALLD 配置 NAT	436
41.11. 丰富规则的优先级	441
41.12. 启用 FIREWALLD 区域中不同接口或源之间的流量转发	444
41.13. 使用 RHEL 系统角色配置 FIREWALLD	446
第 42 章 NFTABLES 入门	453
42.1. 创建和管理 NFTABLES 表、链和规则	453
42.2. 从 IPTABLES 迁移到 NFTABLES	461
42.3. 使用 NFTABLES 配置 NAT	468
42.4. 编写和执行 NFTABLES 脚本	473
42.5. 使用 NFTABLES 命令中的设置	479
42.6. 在 NFTABLES 命令中使用判决映射	483
42.7. 示例：使用 NFTABLES 脚本保护 LAN 和 DMZ	488
42.8. 使用 NFTABLES 限制连接的数量	495
42.9. 调试 NFTABLES 规则	497
42.10. 备份和恢复 NFTABLES 规则集	500
42.11. 其它资源	501
第 43 章 使用 XDP-FILTER 进行高性能流量过滤以防止 DDOS 攻击	502
43.1. 丢弃匹配 XDP-FILTER 规则的网络数据包	502
43.2. 丢弃所有与 XDP-FILTER 规则匹配的网络数据包	504
第 44 章 捕获网络数据包	508
44.1. 使用 XDPDUMP 捕获包括 XDP 程序丢弃的数据包在内的网络数据包	508
44.2. 其他资源	509
第 45 章 了解 RHEL 8 中的 EBPF 网络功能	510
45.1. RHEL 8 中网络 EBPF 功能概述	510
45.2. 按网卡的 RHEL 8 中的 XDP 功能概述	516
第 46 章 使用 BPF 编译器集合进行网络追踪	519
46.1. 安装 BCC-TOOLS 软件包	519
46.2. 显示添加到内核的接受队列中的 TCP 连接	520
46.3. 追踪出去的 TCP 连接尝试	521
46.4. 测量出站 TCP 连接的延迟	521
46.5. 显示被内核丢弃的 TCP 数据包和片段详情	522
46.6. 追踪 TCP 会话	523
46.7. 追踪 TCP 重新传输	524
46.8. 显示 TCP 状态更改信息	525
46.9. 聚合发送到特定子网的 TCP 流量	526
46.10. 通过 IP 地址和端口显示网络吞吐量	528
46.11. 追踪已建立的 TCP 连接	528
46.12. 追踪 IPV4 和 IPV6 倾听尝试	529
46.13. 软中断的服务时间概述	530
46.14. 总结网络接口上的数据包大小和数量	530
第 47 章 配置网络设备以接受来自所有 MAC 地址的流量	533
47.1. 临时配置设备以接受所有流量	533
47.2. 使用 NMCLI 永久配置网络设备，以接受所有流量	534
47.3. 使用 NMSTATECTL 永久配置网络设备，以接受所有流量	535

第 48 章 使用 NMCLI 镜像网络接口	537
第 49 章 使用 NMSTATE-AUTOCONF 自动配置使用 LLDP 的网络状态	539
49.1. 使用 NMSTATE-AUTOCONF 来自动配置网络接口		539
第 50 章 配置 802.3 链路设置	543
50.1. 使用 NMCLI 工具配置 802.3 链接设置		543
第 51 章 DPDK 入门	545
51.1. 安装 DPDK 软件包		545
51.2. 其他资源		545
第 52 章 TIPC 入门	546
52.1. TIPC 的构架		546
52.2. 系统引导时载入 TIPC 模块		547
52.3. 创建 TIPC 网络		548
52.4. 其他资源		549
第 53 章 使用 NM-CLOUD-SETUP 在公有云中自动配置网络接口	551
53.1. 配置和预部署 NM-CLOUD-SETUP		551
53.2. 了解 RHEL EC2 实例中 IMDSV2 角色和 NM-CLOUD-SETUP		553

对红帽文档提供反馈

我们感谢您对我们文档的反馈。让我们了解如何改进它。

通过 Jira 提交反馈（需要帐户）

1. 登录到 [Jira](#) 网站。
2. 单击顶部导航栏中的 **Create**。
3. 在 **Summary** 字段中输入描述性标题。
4. 在 **Description** 字段中输入您对改进的建议。包括文档相关部分的链接。
5. 点对话框底部的 **Create**。

第 1 章 实现一致的网络接口命名

udev 设备管理器在 Red Hat Enterprise Linux 中实现一致的设备命名。设备管理器支持不同的命名方案，默认情况下，根据固件、拓扑和位置信息分配固定的名称。

如果没有一致的设备命名，Linux 内核通过组合固定前缀和索引来为网络接口分配名称。当内核初始化网络设备时，索引会增加。例如：**eth0** 代表启动时探测到的第一个以太网设备。如果您在系统中添加了另一个网络接口控制器，则内核设备名称的分配不再固定，因为重启后，设备可以以不同的顺序初始化。在这种情况下，内核可以以不同的方式命名设备。

要解决这个问题，**udev** 需分配一致的设备名称。它有以下优点：

- 设备名称在重启后是稳定的。
- 即使添加或删除了硬件，设备名称也会保持不变。
- 因此，有问题的硬件可以被无缝地替换。
- 网络命名是无状态的，不需要显式配置文件。

警告



通常，红帽不支持禁用了一致设备命名的系统。有关例外情况，请参阅红帽知识库解决方案 [可以安全地设置 net.ifnames=0 吗](#)。

1.1. UDEV 设备管理器如何重命名网络接口

要为网络接口实现一致的命名方案，**udev** 设备管理器按列出的顺序处理以下规则文件：

1. 可选：`/usr/lib/udev/rules.d/60-net.rules`

`/usr/lib/udev/rules.d/60-net.rules` 文件定义了已弃用的 `/usr/lib/udev/rename_device` 助手工具在 `/etc/sysconfig/network-scripts/ifcfg-*` 文件中搜索 **HWADDR** 参数。如果变量中设置的值与接口的 MAC 地址匹配，则助手工具会将接口重命名为 `ifcfg` 文件的 **DEVICE** 参数中设置的名称。

如果系统只使用 keyfile 格式的 NetworkManager 连接配置文件，**udev** 会跳过这一步。

2. 只在 Dell 系统上：`/usr/lib/udev/rules.d/71-biosdevname.rules`

只有安装了 **biosdevname** 软件包，且规则文件定义了 **biosdevname** 工具根据其命名策略重命名了接口，如果其没有在之前的步骤中被重命名，该文件才存在。



注意

仅在 Dell 系统上安装和使用 **biosdevname**。

3. `/usr/lib/udev/rules.d/75-net-description.rules`

此文件定义了 **udev** 如何检查网络接口，并在 **udev-internal** 变量中设置属性。然后，这些变量由 `/usr/lib/udev/rules.d/80-net-setup-link.rules` 文件在下一步中进行处理。其中一些属性可以未定义。

4. /usr/lib/udev/rules.d/80-net-setup-link.rules

此文件调用 **udev** 服务的 **net_setup_link** builtin，并且 **udev** 根据 **/usr/lib/systemd/network/99-default.link** 文件中 **NamePolicy** 参数中策略的顺序重命名接口。详情请查看 [网络接口命名策略](#)。

如果没有应用任何策略，则 **udev** 不会重命名接口。

其他资源

- [为什么 systemd 网络接口名称在主 RHEL 版本之间有所不同](#) (红帽知识库)

1.2. 网络接口命名策略

默认情况下，**udev** 设备管理器使用 **/usr/lib/systemd/network/99-default.link** 文件来确定在重命名接口时要应用哪些设备命名策略。此文件中的 **NamePolicy** 参数定义 **udev** 使用哪些策略以及使用哪个顺序：

NamePolicy=kernel database onboard slot path

根据哪个策略与 **NamePolicy** 参数指定的首先匹配，下表描述了 **udev** 的不同操作：

policy	描述	名称示例
kernel	如果内核表示设备名称是可预测的，则 udev 不会重命名这个设备。	lo
database	此策略根据 udev 硬件数据库中的映射分配名称。详情请查看您系统上的 hwdb (7) 手册页。	idrac
onboard	设备名称包含固件或者 BIOS 提供的索引号，用于板上的设备。	eno1
slot	设备名称包含固件或 BIOS 提供的 PCI Express (PCIe)热插拔插槽索引号。	ens1
path	设备名称包含硬件连接器的物理位置。	enp1s0
mac	设备名称包含 MAC 地址。默认情况下，Red Hat Enterprise Linux 不使用此策略，但管理员可以启用它。	enx525400d5e0fb

其他资源

- [udev 设备管理器如何重命名网络接口](#)
- 您系统上的 **systemd.link (5)** 手册页

1.3. 网络接口命名方案

udev 设备管理器使用设备驱动程序提供的某些稳定的接口属性，来生成一致的设备名称。

如果新的 **udev** 版本更改了服务如何为某些接口创建名称的方式，红帽会添加一个新的方案版本，并在您系统上的 **systemd.net-naming-scheme (7)** 手册页中记录详细信息。默认情况下，Red Hat Enterprise Linux (RHEL) 8 使用 **rhel-8.0** 命名方案，即使您安装或升级到了更新的 RHEL 次版本。

如果要使用默认方案以外的方案，您可以 [切换网络接口命名方案](#)。

有关不同设备类型和平台的命名方案的详情，请查看您系统上的 **systemd.net-naming-scheme (7)** 手册页。

1.4. 切换到不同的网络接口命名方案

默认情况下，Red Hat Enterprise Linux (RHEL) 8 使用 **rhel-8.0** 命名方案，即使您安装或升级到了更新的 RHEL 次版本。虽然默认的命名方案适合大多数情况，但可能有切换到不同的方案版本的理由，例如：

- 如果其向接口名称添加了额外的属性（如插槽号），则新方案可以帮助更好地识别设备。
- 新方案可以防止 **udev** 回退到内核分配的设备名称(**eth***)。如果驱动程序没有为两个或多个接口提供足够的唯一属性，来为它们生成唯一名称，则会发生这种情况。

先决条件

- 您可以访问服务器的控制台。

步骤

- 列出网络接口：

```
# ip link show
2: eno1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
mode DEFAULT group default qlen 1000
    link/ether 00:00:5e:00:53:1a brd ff:ff:ff:ff:ff:ff
    ...
...
```

记录接口的 MAC 地址。

- 可选：显示网络接口的 **ID_NET_NAMING_SCHEME** 属性，来识别 RHEL 当前使用的命名方案：

```
# udevadm info --query=property /sys/class/net/eno1 | grep
"ID_NET_NAMING_SCHEME"
ID_NET_NAMING_SCHEME=rhel-8.0
```

请注意，属性在 **lo** loopback 设备上不可用。

- 将 **net.naming-scheme=<scheme>** 选项附加到所有安装的内核的命令行中，例如：

```
# grubby --update-kernel=ALL --args=net.naming-scheme=rhel-8.4
```

- 重启系统。

```
# reboot
```

- 根据您记录的 MAC 地址，识别因不同的命名方案而更改的网络接口的新名称：

```
# ip link show
```

```
2: eno1np0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen 1000
    link/ether 00:00:5e:00:53:1a brd ff:ff:ff:ff:ff:ff
    ...

```

切换方案后，**udev** 将具有指定 MAC 地址 **eno1np0** 的设备命名为 **eno1**，而之前被命名为 **eno1**。

6. 确定哪个 NetworkManager 连接配置文件使用之前名称的接口：

```
# nmcli -f device,name connection show
DEVICE NAME
eno1 example_profile
...
```

7. 将连接配置文件中的 **connection.interface-name** 属性设置为新接口名称：

```
# nmcli connection modify example_profile connection.interface-name "eno1np0"
```

8. 重新激活连接配置文件：

```
# nmcli connection up example_profile
```

验证

- 通过显示网络接口的 **ID_NET_NAMING_SCHEME** 属性来识别 RHEL 现在使用的命名方案：

```
# udevadm info --query=property /sys/class/net/eno1np0 | grep
"ID_NET_NAMING_SCHEME"
ID_NET_NAMING_SCHEME=_rhel-8.4
```

其他资源

- [网络接口命名方案](#)

1.5. 在 IBM Z 平台上确定可预测的 ROCE 设备名称

在 Red Hat Enterprise Linux (RHEL) 8.7 及更新版本上，**udev** 设备管理器为 IBM Z 上的 RoCE 接口设置名称，如下所示：

- 如果主机对设备强制唯一标识符(UID)，则 **udev** 会分配一个基于 UID 的一致的设备名称，如 **eno<UID_in_decimal>**。
- 如果主机没有为设备强制 UID，则行为取决于您的设置：
 - 默认情况下，**udev** 为设备使用无法预料的名称。
 - 如果您设置了 **net.naming-scheme=rhel-8.7** 内核命令行选项，则 **udev** 会分配一个基于设备功能标识符(FID)的一致的设备名称，例如 **ens<fid_in_decimal>**。

在以下情况下，为 IBM Z 上的 RoCE 接口手动配置可预测的设备名称：

- 您的主机运行 RHEL 8.6 或更早版本，并对设备强制 UID，并计划更新到 RHEL 8.7 或更高版本。

升级到 RHEL 8.7 或更高版本后，**udev** 使用一致的接口名称。但是，如果您在更新前使用了无法预料的设备名称，NetworkManager 连接配置文件仍然使用这些名称，且不能激活，直到您更新了受影响的配置文件。

- 您的主机运行 RHEL 8.7 或更高版本，且不强制 UID，您计划升级到 RHEL 9。

在使用 **udev** 规则或 **systemd** 链接文件手动重命名接口前，您必须确定可预测的设备名称。

先决条件

- RoCE 控制器已安装在系统上。
- **sysfsutils** 软件包已安装。

流程

1. 显示可用的网络设备，并记录 RoCE 设备的名称：

```
# ip link show
...
2: enP5165p0s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000
...
...
```

2. 显示 **/sys** 文件系统中的设备路径：

```
# systool -c net -p
Class = "net"

Class Device = "enP5165p0s0"
Class Device path = "/sys/devices/pci142d:00/142d:00:00.0/net/enP5165p0s0"
Device = "142d:00:00.0"
Device path = "/sys/devices/pci142d:00/142d:00:00.0"
```

在下一步中使用 **Device path** 字段中显示的路径。

3. 显示 **<device_path>/uid_id_unique** 文件的值，例如：

```
# cat /sys/devices/pci142d:00/142d:00:00.0/uid_id_unique
```

显示的值指示 UID 唯一性是否是强制的，并且您在以后的步骤中需要这个值。

4. 确定唯一标识符：

- 如果 UID 唯一性是强制的(1)，显示存储在 **<device_path>/uid** 文件中的 UID，例如：

```
# cat /sys/devices/pci142d:00/142d:00:00.0/uid
```

- 如果 UID 唯一性不是强制的(0)，显示存储在 **<device_path>/function_id** 文件中的 FID，例如：

```
# cat /sys/devices/pci142d:00/142d:00:00.0/function_id
```

命令的输出显示十六进制的 UID 和 FID 值。

5. 将十六进制标识符转换为十进制，例如：

```
# printf "%d\n" 0x00001402
5122
```

6. 要确定可预测的设备名称，根据 UID 唯一性是否是强制的，将十进制格式的标识符附加到相应的前缀：

- 如果 UID 唯一性是强制的，请将标识符附加到 **eno** 前缀，例如 **eno5122**。
- 如果没有强制 UID 唯一性，请将标识符附加到 **ens** 前缀，如 **ens5122**。

后续步骤

- 使用以下方法之一将接口重命名为可预测的名称：
 - 使用 **udev** 规则配置用户定义的网络接口名称
 - 使用 **systemd** 链接文件配置用户定义的网络接口名称

其他资源

- IBM 文档：[网络接口名称](#)
- 您系统上的 **systemd.net-naming-scheme (7)** 手册页

1.6. 在安装过程中为以太网接口自定义前缀

如果您不想将默认 device-naming 策略用于以太网接口，您可以在 Red Hat Enterprise Linux (RHEL) 安装过程中设置一个自定义设备前缀。



重要

只有您在 RHEL 安装过程中设置了前缀，红帽才支持带有自定义以太网前缀的系统。不支持在已部署的系统上使用 **prefixdevname** 工具。

如果您在安装过程中设置了设备前缀，则 **udev** 服务在安装后对以太网接口使用 **<prefix><index>** 格式。例如，如果您设置了前缀 **net**，服务会将名称 **net0**、**net1** 等分配给以太网接口。

udev 服务将索引附加到自定义前缀，并保留已知以太网接口的索引值。如果您添加一个接口，**udev** 会为新接口分配一个比之前分配的索引值大的索引值。

先决条件

- 前缀由 ASCII 字符组成。
- 前缀是一个字母数字字符串。
- 前缀少于 16 个字符。
- 前缀不会与任何其他熟知的网络接口前缀冲突，如 **eth**、**eno**、**ens** 和 **em**。

步骤

1. 引导 Red Hat Enterprise Linux 安装介质。
2. 在引导管理器中, 请按照以下步骤操作 :
 - a. 选择 **Install Red Hat Enterprise Linux <version>** 条目。
 - b. 按 **Tab** 编辑条目。
 - c. 将 **net.ifnames.prefix=<prefix>** 追加到在内核选项中。
 - d. 按 **Enter** 启动安装程序。
3. 安装 Red Hat Enterprise Linux。

验证

- 要验证接口名称, 请显示网络接口 :

```
# ip link show
...
2: net0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
mode DEFAULT group default qlen 1000
    link/ether 00:00:5e:00:53:1a brd ff:ff:ff:ff:ff
...
...
```

其他资源

- [从安装介质以交互方式安装 RHEL](#)

1.7. 使用 UDEV 规则配置用户定义的网络接口名称

您可以使用 **udev** 规则来实现反映您机构要求的自定义网络接口名称。

流程

1. 识别您要重命名的网络接口 :

```
# ip link show
...
enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
mode DEFAULT group default qlen 1000
    link/ether 00:00:5e:00:53:1a brd ff:ff:ff:ff:ff
...
...
```

记录接口的 MAC 地址。

2. 显示接口的设备类型 ID :

```
# cat /sys/class/net/enp1s0/type
1
```

3. 创建 **/etc/udev/rules.d/70-persistent-net.rules** 文件, 并为您要重命名的每个接口添加一个规则 :

```
SUBSYSTEM=="net",ACTION=="add",ATTR{address}=="<MAC_address>",ATTR{type}=="<device_type_id>",NAME=="<new_interface_name>"
```



重要

如果您在引导过程中需要一致的设备名称，则只使用 **70-persistent-net.rules** 作为文件名。如果您重新生成 RAM 磁盘镜像，则 **dracut** 工具会在 **initrd** 镜像中添加具有此名称的文件。

例如，使用以下规则将 MAC 地址为 **00:00:5e:00:53:1a** 的接口重命名为 **provider0**：

```
SUBSYSTEM=="net",ACTION=="add",ATTR{address}=="00:00:5e:00:53:1a",ATTR{type}=="1",NAME=="provider0"
```

4. 可选：重新生成 **initrd** RAM 磁盘镜像：

```
# dracut -f
```

只有在 RAM 磁盘中需要网络功能时才需要这一步。例如，如果 root 文件系统存储在网络设备上，如 iSCSI，则是这种情况。

5. 确定哪个 NetworkManager 连接配置文件使用您要重命名的接口：

```
# nmcli -f device,name connection show
DEVICE  NAME
enp1s0  example_profile
...
```

6. 在连接配置文件中取消 **connection.interface-name** 属性的设置：

```
# nmcli connection modify example_profile connection.interface-name ""
```

7. 临时配置连接配置文件，以匹配新的和以前的接口名称：

```
# nmcli connection modify example_profile match.interface-name "provider0 enp1s0"
```

8. 重启系统：

```
# reboot
```

9. 验证具有您在链接文件中指定的 MAC 地址的设备已重命名为 **provider0**：

```
# ip link show
provider0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode
DEFAULT group default qlen 1000
    link/ether 00:00:5e:00:53:1a brd ff:ff:ff:ff:ff:ff
...
```

10. 将连接配置文件配置为仅匹配新接口名称：

```
# nmcli connection modify example_profile match.interface-name "provider0"
```

现在，您已从连接配置文件中删除了旧接口名称。

11. 重新激活连接配置文件：

```
# nmcli connection up example_profile
```

其他资源

- 您系统上的 **udev (7)** 手册页

1.8. 使用 **SYSTEMD** 链接文件配置用户定义的网络接口名称

您可以使用 **systemd** 链接文件来实现反映您机构要求的自定义网络接口名称。

先决条件

- 您必须满足以下条件之一：NetworkManager 不管理这个接口，或者使用 [keyfile 格式](#) 的相应的连接配置文件。

流程

1. 识别您要重命名的网络接口：

```
# ip link show
...
enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
mode DEFAULT group default qlen 1000
    link/ether 00:00:5e:00:53:1a brd ff:ff:ff:ff:ff:ff
...
...
```

记录接口的 MAC 地址。

2. 如果不存在，请创建 **/etc/systemd/network/** 目录：

```
# mkdir -p /etc/systemd/network/
```

3. 对于您要重命名的每个接口，请在 **/etc/systemd/network/** 目录中，使用如下内容创建一个 **70-*.link** 文件：

```
[Match]
MACAddress=<MAC_address>

[Link]
Name=<new_interface_name>
```



重要

使用具有 **70-** 前缀的文件名，使文件名与基于 **udev** 规则的解决方案一致。

例如，使用以下内容创建 **/etc/systemd/network/70-provider0.link** 文件，将 MAC 地址为 **00:00:5e:00:53:1a** 的接口重命名为 **provider0**：

```
[Match]
MACAddress=00:00:5e:00:53:1a

[Link]
Name=provider0
```

4. 可选：重新生成 **initrd** RAM 磁盘镜像：

```
# dracut -f
```

只有在 RAM 磁盘中需要网络功能时才需要这一步。例如，如果 root 文件系统存储在网络设备上，如 iSCSI，则是这种情况。

5. 确定哪个 NetworkManager 连接配置文件使用您要重命名的接口：

```
# nmcli -f device,name connection show
DEVICE NAME
enp1s0 example_profile
...
```

6. 在连接配置文件中取消 **connection.interface-name** 属性的设置：

```
# nmcli connection modify example_profile connection.interface-name ""
```

7. 临时配置连接配置文件，以匹配新的和以前的接口名称：

```
# nmcli connection modify example_profile match.interface-name "provider0 enp1s0"
```

8. 重启系统：

```
# reboot
```

9. 验证具有您在链接文件中指定的 MAC 地址的设备已重命名为 **provider0**：

```
# ip link show
provider0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode
DEFAULT group default qlen 1000
    link/ether 00:00:5e:00:53:1a brd ff:ff:ff:ff:ff:ff
...
```

10. 将连接配置文件配置为仅匹配新接口名称：

```
# nmcli connection modify example_profile match.interface-name "provider0"
```

现在，您已从连接配置文件中删除了旧接口名称。

11. 重新激活连接配置文件。

```
# nmcli connection up example_profile
```

- 您系统上的 **systemd.link (5)** 手册页

1.9. 使用 SYSTEMD 链接文件将替代名称分配给网络接口

通过替代接口命名，内核可以将额外名称分配给网络接口。您可以使用这些替代名称，方式与需要网络接口名称的命令中的普通接口名称相同。

先决条件

- 您必须对替代名称使用 ASCII 字符。
- 备用名称必须小于 128 个字符。

流程

1. 显示网络接口名称及其 MAC 地址：

```
# ip link show
...
enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
mode DEFAULT group default qlen 1000
    link/ether 00:00:5e:00:53:1a brd ff:ff:ff:ff:ff:ff
...
...
```

记录您要为其分配替代名称的接口的 MAC 地址。

2. 如果不存在，请创建 **/etc/systemd/network/** 目录：

```
# mkdir -p /etc/systemd/network/
```

3. 对于必须有一个备用名称的每个接口，在 **/etc/systemd/network/** 目录中创建一个具有唯一名称和 **.link** 后缀的 **/usr/lib/systemd/network/99-default.link** 文件的副本，例如：

```
# cp /usr/lib/systemd/network/99-default.link /etc/systemd/network/98-lan.link
```

4. 修改您在上一步中创建的文件。按如下所示重写 **[Match]** 部分，并将 **AlternativeName** 条目附加到 **[Link]** 部分：

```
[Match]
MACAddress=<MAC_address>

[Link]
...
AlternativeName=<alternative_interface_name_1>
AlternativeName=<alternative_interface_name_2>
AlternativeName=<alternative_interface_name_n>
```

例如，使用以下内容创建 **/etc/systemd/network/70-altname.link** 文件，来将 **provider** 作为替代名称分配给 MAC 地址为 **00:00:5e:00:53:1a** 的接口：

```
[Match]
MACAddress=00:00:5e:00:53:1a
```

[Link]

```
NamePolicy=kernel database onboard slot path  
AlternativeNamesPolicy=database onboard slot path  
MACAddressPolicy=persistent  
AlternativeName=provider
```

5. 重新生成 **initrd** RAM 磁盘镜像：

```
# dracut -f
```

6. 重启系统：

```
# reboot
```

验证

- 使用替代接口名称。例如，显示具有替代名称 **provider** 的设备的 IP 地址设置：

```
# ip address show provider  
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP  
group default qlen 1000  
    link/ether 00:00:5e:00:53:1a brd ff:ff:ff:ff:ff:ff  
    altname provider  
    ...
```

其他资源

- [接口命名方案中的 alternativesNamesPolicy 是什么？（红帽知识库）](#)

第 2 章 配置以太网连接

NetworkManager 为主机上安装的每个以太网适配器创建一个连接配置文件。默认情况下，此配置文件将 DHCP 用于 IPv4 和 IPv6 连接。修改此自动创建的配置文件，或在以下情况下添加新配置文件：

- 网络需要自定义设置，如静态 IP 地址配置。
- 您需要多个配置文件，因为主机在不同的网络中漫游。

Red Hat Enterprise Linux 为管理员提供不同的选项来配置以太网连接。例如：

- 在命令行中使用 **nmcli** 配置连接。
- 使用 **nmtui** 在基于文本的用户界面中配置连接。
- 使用 GNOME Settings 菜单或 **nm-connection-editor** 应用程序在图形界面中配置连接。
- 使用 **nmstatectl** 通过 **nmstate** API 配置连接。
- 使用 RHEL 系统角色自动化一个或多个主机上连接的配置。



注意

如果要对运行在 Microsoft Azure 云中的主机手动配置以太网连接，请禁用 **cloud-init** 服务或将其配置为忽略从云环境检索到的网络设置。否则，**cloud-init** 将在下次重启时覆盖您手动配置的网络设置。

2.1. 使用 NMCLI 配置以太网连接

如果您通过以太网将主机连接到网络，您可以使用 **nmcli** 工具在命令行上管理连接的设置。

先决条件

- 服务器配置中存在物理或虚拟以太网网络接口控制器(NIC)。

步骤

1. 列出 NetworkManager 连接配置文件：

```
# nmcli connection show
NAME           UUID             TYPE      DEVICE
Wired connection 1  a5eb6490-cc20-3668-81f8-0314a27f3f75  ethernet  enp1s0
```

默认情况下，NetworkManager 为主机中的每个 NIC 创建一个配置文件。如果您计划仅将这个 NIC 连接到特定的网络，请调整自动创建的配置文件。如果您计划使用不同的设置将这个 NIC 连接到网络，请为每个网络创建单独的配置文件。

2. 如果要创建额外的连接配置文件，请输入：

```
# nmcli connection add con-name <connection-name> ifname <device-name> type
ethernet
```

跳过此步骤来修改现有的配置文件。

3. 可选：重命名连接配置文件：

```
# nmcli connection modify "Wired connection 1" connection.id "Internal-LAN"
```

在有多个配置文件的主机上，有意义的名称可以更容易识别配置文件的用途。

4. 显示连接配置文件的当前设置：

```
# nmcli connection show Internal-LAN
...
connection.interface-name:    enp1s0
connection.autoconnect:      yes
ipv4.method:                  auto
ipv6.method:                  auto
...
```

5. 配置 IPv4 设置：

- 要使用 DHCP，请输入：

```
# nmcli connection modify Internal-LAN ipv4.method auto
```

如果 **ipv4.method** 已设置为 **auto**（默认），请跳过这一步。

- 要设置静态 IPv4 地址、网络掩码、默认网关、DNS 服务器和搜索域，请输入：

```
# nmcli connection modify Internal-LAN ipv4.method manual ipv4.addresses
192.0.2.1/24 ipv4.gateway 192.0.2.254 ipv4.dns 192.0.2.200 ipv4.dns-search
example.com
```

6. 配置 IPv6 设置：

- 要使用无状态地址自动配置(SLAAC)，请输入：

```
# nmcli connection modify Internal-LAN ipv6.method auto
```

如果 **ipv6.method** 已设置为 **auto**（默认），请跳过这一步。

- 要设置静态 IPv6 地址、网络掩码、默认网关、DNS 服务器和搜索域，请输入：

```
# nmcli connection modify Internal-LAN ipv6.method manual ipv6.addresses
2001:db8:1::ffffe/64 ipv6.gateway 2001:db8:1::ffffe ipv6.dns 2001:db8:1::ffbb
ipv6.dns-search example.com
```

7. 要在配置文件中自定义其他设置，请使用以下命令：

```
# nmcli connection modify <connection-name> <setting> <value>
```

用空格或分号将值括起来。

8. 激活配置文件：

```
# nmcli connection up Internal-LAN
```

验证

- 显示 NIC 的 IP 设置：

```
# ip address show enp1s0
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
group default qlen 1000
    link/ether 52:54:00:17:b8:b6 brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.1/24 brd 192.0.2.255 scope global noprefixroute enp1s0
        valid_lft forever preferred_lft forever
    inet6 2001:db8:1::ffe/64 scope global noprefixroute
        valid_lft forever preferred_lft forever
```

- 显示 IPv4 默认网关：

```
# ip route show default
default via 192.0.2.254 dev enp1s0 proto static metric 102
```

- 显示 IPv6 默认网关：

```
# ip -6 route show default
default via 2001:db8:1::ffe dev enp1s0 proto static metric 102 pref medium
```

- 显示 DNS 设置：

```
# cat /etc/resolv.conf
search example.com
nameserver 192.0.2.200
nameserver 2001:db8:1::ffbb
```

如果多个连接配置文件同时处于活跃状态，则 **nameserver** 条目的顺序取决于这些配置文件中的 DNS 优先级值和连接类型。

- 使用 **ping** 工具验证这个主机是否可以向其他主机发送数据包：

```
# ping <host-name-or-IP-address>
```

故障排除

- 验证网线是否插入到主机和交换机。
- 检查链路失败是否只存在于此主机上，或者连接到同一交换机的其它主机上。
- 验证网络电缆和网络接口是否如预期工作。执行硬件诊断步骤，并替换有问题的网线和网络接口卡。
- 如果磁盘中的配置与设备中的配置不匹配，则启动或重启 NetworkManager 会创建一个代表该设备的配置的内存连接。有关详情以及如何避免此问题，请参阅红帽知识库解决方案 [NetworkManager 在重启 NetworkManager 服务后复制连接](#)。

其他资源

- 您系统上的 **nm-settings (5)** 手册页

2.2. 使用 **NMCLI** 交互式编辑器配置以太网连接

如果您通过以太网将主机连接到网络，您可以使用 **nmcli** 工具在命令行上管理连接的设置。

先决条件

- 服务器配置中存在物理或虚拟以太网网络接口控制器(NIC)。

步骤

- 列出 NetworkManager 连接配置文件：

```
# nmcli connection show
NAME           UUID            TYPE      DEVICE
Wired connection 1  a5eb6490-cc20-3668-81f8-0314a27f3f75  ethernet  enp1s0
```

默认情况下，NetworkManager 为主机中的每个 NIC 创建一个配置文件。如果您计划仅将这个 NIC 连接到特定的网络，请调整自动创建的配置文件。如果您计划使用不同的设置将这个 NIC 连接到网络，请为每个网络创建单独的配置文件。

- 以互动模式启动 **nmcli**：

- 要创建额外的连接配置文件，请输入：

```
# nmcli connection edit type ethernet con-name "<connection-name>"
```

- 要修改现有的连接配置文件，请输入：

```
# nmcli connection edit con-name "<connection-name>"
```

- 可选：重命名连接配置文件：

```
nmcli> set connection.id Internal-LAN
```

在有多个配置文件的主机上，有意义的名称可以更容易识别配置文件的用途。

不要使用引号来设置包含空格的 ID，以避免 **nmcli** 将引号作为名称的一部分。例如，要将 **Example Connection** 设置为 ID，请输入 **set connection.id Example Connection**。

- 显示连接配置文件的当前设置：

```
nmcli> print
...
connection.interface-name:  enp1s0
connection.autoconnect:    yes
ipv4.method:              auto
ipv6.method:              auto
...
```

- 如果创建一个新的连接配置文件，请设置网络接口：

```
nmcli> set connection.interface-name enp1s0
```

6. 配置 IPv4 设置：

- 要使用 DHCP，请输入：

```
nmcli> set ipv4.method auto
```

如果 **ipv4.method** 已设置为 **auto**（默认），请跳过这一步。

- 要设置静态 IPv4 地址、网络掩码、默认网关、DNS 服务器和搜索域，请输入：

```
nmcli> ipv4.addresses 192.0.2.1/24
Do you also want to set 'ipv4.method' to 'manual'? [yes]: yes
nmcli> ipv4.gateway 192.0.2.254
nmcli> ipv4.dns 192.0.2.200
nmcli> ipv4.dns-search example.com
```

7. 配置 IPv6 设置：

- 要使用无状态地址自动配置(SLAAC)，请输入：

```
nmcli> set ipv6.method auto
```

如果 **ipv6.method** 已设置为 **auto**（默认），请跳过这一步。

- 要设置静态 IPv6 地址、网络掩码、默认网关、DNS 服务器和搜索域，请输入：

```
nmcli> ipv6.addresses 2001:db8:1::fffe/64
Do you also want to set 'ipv6.method' to 'manual'? [yes]: yes
nmcli> ipv6.gateway 2001:db8:1::fffe
nmcli> ipv6.dns 2001:db8:1::ffbb
nmcli> ipv6.dns-search example.com
```

8. 保存并激活连接：

```
nmcli> save persistent
```

9. 保留为互动模式：

```
nmcli> quit
```

验证

1. 显示 NIC 的 IP 设置：

```
# ip address show enp1s0
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
group default qlen 1000
    link/ether 52:54:00:17:b8:b6 brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.1/24 brd 192.0.2.255 scope global noprefixroute enp1s0
        valid_lft forever preferred_lft forever
    inet6 2001:db8:1::fffe/64 scope global noprefixroute
        valid_lft forever preferred_lft forever
```

2. 显示 IPv4 默认网关：

```
# ip route show default
default via 192.0.2.254 dev enp1s0 proto static metric 102
```

3. 显示 IPv6 默认网关：

```
# ip -6 route show default
default via 2001:db8:1::ffffe dev enp1s0 proto static metric 102 pref medium
```

4. 显示 DNS 设置：

```
# cat /etc/resolv.conf
search example.com
nameserver 192.0.2.200
nameserver 2001:db8:1::ffbb
```

如果多个连接配置文件同时处于活跃状态，则 **nameserver** 条目的顺序取决于这些配置文件中的 DNS 优先级值和连接类型。

5. 使用 **ping** 工具验证这个主机是否可以向其他主机发送数据包：

```
# ping <host-name-or-IP-address>
```

故障排除

- 验证网线是否插入到主机和交换机。
- 检查链路失败是否只存在于此主机上，或者连接到同一交换机的其它主机上。
- 验证网络电缆和网络接口是否如预期工作。执行硬件诊断步骤，并替换有问题的网线和网络接口卡。
- 如果磁盘中的配置与设备中的配置不匹配，则启动或重启 NetworkManager 会创建一个代表该设备的配置的内存连接。有关详情以及如何避免此问题，请参阅红帽知识库解决方案 [NetworkManager 在重启 NetworkManager 服务后复制连接](#)。

其他资源

- 您系统上的 **nm-settings (5)** 和 **nmcli (1)** 手册页

2.3. 使用 NMTUI 配置以太网连接

如果通过以太网将主机连接到网络，您可以使用 **nmtui** 应用程序在基于文本的用户界面中管理连接的设置。使用 **nmtui** 创建新配置文件，并在没有图形界面的主机上更新现有配置文件。



注意

在 **nmtui** 中：

- 使用光标键导航。
- 选择一个按钮并按 **Enter** 键。
- 使用 **Space** 选择并清除复选框。
- 要返回上一个屏幕，请使用 **ESC**。

先决条件

- 服务器配置中存在物理或虚拟以太网网络接口控制器(NIC)。

流程

1. 如果您不知道连接中使用的网络设备名称，显示可用的设备：

```
# nmcli device status
DEVICE  TYPE      STATE           CONNECTION
enp1s0   ethernet  unavailable    --
...
...
```

2. 启动 **nmtui**：

```
# nmtui
```

3. 选择 **Edit a connection**，然后按 **Enter**。

4. 选择是否添加新连接配置文件或修改现有连接配置文件：

- 要创建新配置文件：

- i. 按 **Add**。
- ii. 从网络类型列表中选择 **Ethernet**，然后按 **Enter**。

- 要修改现有的配置文件，请从列表中选择配置文件，然后按 **Enter**。

5. 可选：更新连接配置文件的名称。

在有多个配置文件的主机上，有意义的名称可以更容易识别配置文件的用途。

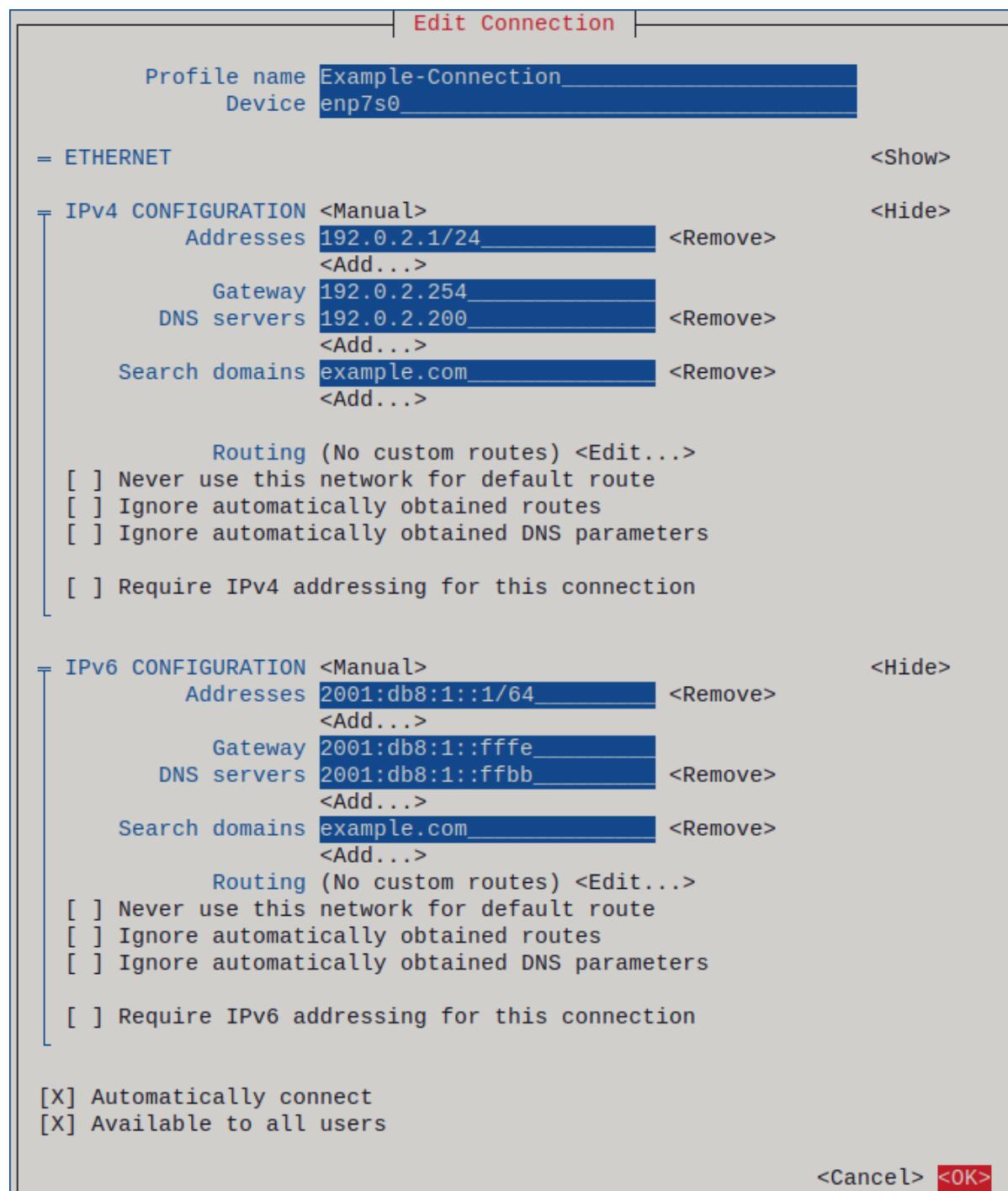
6. 如果创建新连接配置文件，请在 **Device** 字段中输入网络设备名称。

7. 根据您的环境，相应地在 **IPv4 configuration** 和 **IPv6 configuration** 区中配置 IP 地址。为此，请按这些区域旁边的按钮，并选择：

- **Disabled**，如果此连接不需要 IP 地址。
- **Automatic**，如果 DHCP 服务器动态为这个 NIC 分配一个 IP 地址。
- **Manual**，如果网络需要静态 IP 地址设置。在这种情况下，您必须填写更多字段：
 - i. 按您要配置的协议旁边的 **Show** 来显示其他字段。

- ii. 按 **Addresses** 旁边的 **Add**, 并输入无类别域间路由(CIDR)格式的 IP 地址和子网掩码。如果没有指定子网掩码, NetworkManager 会为 IPv4 地址设置 /32 子网掩码, 并为 IPv6 地址设置 /64。
- iii. 输入默认网关的地址。
- iv. 按 **DNS servers** 旁边的 **Add**, 并输入 DNS 服务器地址。
- v. 按 **Search domains** 旁边的 **Add**, 并输入 DNS 搜索域。

图 2.1. 使用静态 IP 地址设置的以太网连接示例



8. 按 **OK** 创建并自动激活新连接。
9. 按 **Back** 返回到主菜单。

- 选择 **Quit**, 然后按 **Enter** 键关闭 **nmtui** 应用程序。

验证

- 显示 NIC 的 IP 设置：

```
# ip address show enp1s0
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
group default qlen 1000
    link/ether 52:54:00:17:b8:b6 brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.1/24 brd 192.0.2.255 scope global noprefixroute enp1s0
        valid_lft forever preferred_lft forever
    inet6 2001:db8:1::fffe/64 scope global noprefixroute
        valid_lft forever preferred_lft forever
```

- 显示 IPv4 默认网关：

```
# ip route show default
default via 192.0.2.254 dev enp1s0 proto static metric 102
```

- 显示 IPv6 默认网关：

```
# ip -6 route show default
default via 2001:db8:1::ffffe dev enp1s0 proto static metric 102 pref medium
```

- 显示 DNS 设置：

```
# cat /etc/resolv.conf
search example.com
nameserver 192.0.2.200
nameserver 2001:db8:1::ffbb
```

如果多个连接配置文件同时处于活跃状态，则 **nameserver** 条目的顺序取决于这些配置文件中的 DNS 优先级值和连接类型。

- 使用 **ping** 工具验证这个主机是否可以向其他主机发送数据包：

```
# ping <host-name-or-IP-address>
```

故障排除

- 验证网线是否插入到主机和交换机。
- 检查链路失败是否只存在于此主机上，或者连接到同一交换机的其它主机上。
- 验证网络电缆和网络接口是否如预期工作。执行硬件诊断步骤，并替换有问题的网线和网络接口卡。
- 如果磁盘中的配置与设备中的配置不匹配，则启动或重启 NetworkManager 会创建一个代表该设备的配置的内存连接。有关详情以及如何避免此问题，请参阅红帽知识库解决方案 [NetworkManager 在重启 NetworkManager 服务后复制连接](#)。

其他资源

- 配置 NetworkManager 以避免使用特定配置集提供默认网关
- 配置 DNS 服务器顺序

2.4. 使用控制中心配置以太网连接

如果您通过以太网将主机连接到网络，您可以使用 GNOME Settings 菜单，通过图形界面管理连接的设置。

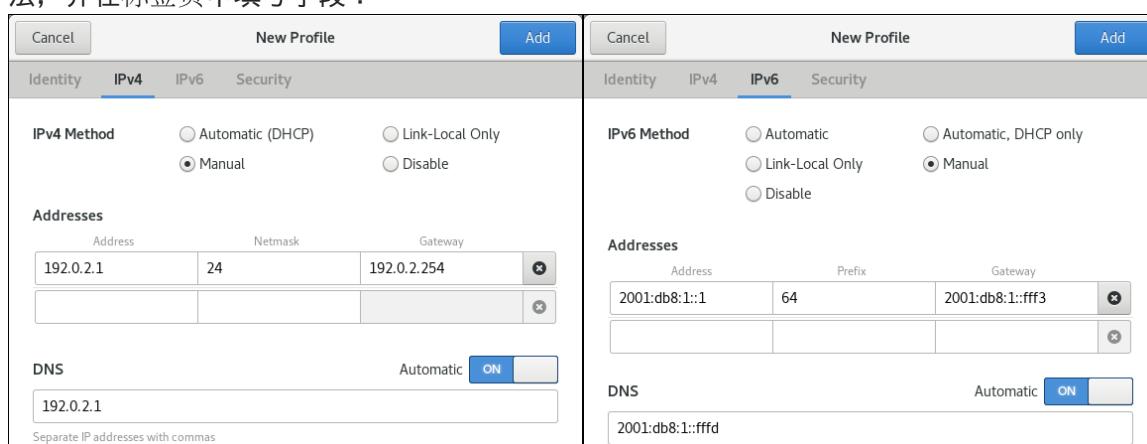
请注意，**control-center** 不支持与 **nm-connection-editor** 应用程序或 **nmcli** 实用程序一样多的配置选项。

先决条件

- 服务器配置中存在物理或虚拟以太网网络接口控制器(NIC)。
- 已安装了 GNOME。

步骤

1. 按 **Super** 键，输入 **Settings**，然后按 **Enter** 键。
2. 在左侧导航中选择 **Network**。
3. 选择是否添加新连接配置文件或修改现有连接配置文件：
 - 要创建新配置文件，请单击 **Ethernet** 条目旁边的 **+** 按钮。
 - 要修改现有配置文件，请点击配置文件条目旁的齿轮图标。
4. 可选：在 **Identity** 选项卡中，更新连接配置文件的名称。
在有多个配置文件的主机上，有意义的名称可以更容易识别配置文件的用途。
5. 根据您的环境，相应地在 **IPv4** 和 **IPv6** 标签页中配置 IP 地址设置：
 - 要使用 DHCP 或 IPv6 无状态地址自动配置(SLAAC)，请选择 **Automatic (DHCP)** 作为方法（默认）。
 - 要设置静态 IP 地址、网络掩码、默认网关、DNS 服务器和搜索域，请选择 **Manual** 作为方法，并在标签页中填写字段：



6. 根据您是否添加或修改连接配置文件，点 **Add** 或 **Apply** 按钮保存连接。
GNOME **control-center** 会自动激活连接。

验证

- 显示 NIC 的 IP 设置：

```
# ip address show enp1s0
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
group default qlen 1000
    link/ether 52:54:00:17:b8:b6 brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.1/24 brd 192.0.2.255 scope global noprefixroute enp1s0
        valid_lft forever preferred_lft forever
    inet6 2001:db8:1::ffe/64 scope global noprefixroute
        valid_lft forever preferred_lft forever
```

- 显示 IPv4 默认网关：

```
# ip route show default
default via 192.0.2.254 dev enp1s0 proto static metric 102
```

- 显示 IPv6 默认网关：

```
# ip -6 route show default
default via 2001:db8:1::ffe dev enp1s0 proto static metric 102 pref medium
```

- 显示 DNS 设置：

```
# cat /etc/resolv.conf
search example.com
nameserver 192.0.2.200
nameserver 2001:db8:1::ffbb
```

如果多个连接配置文件同时处于活跃状态，则 **nameserver** 条目的顺序取决于这些配置文件中的 DNS 优先级值和连接类型。

- 使用 **ping** 工具验证这个主机是否可以向其他主机发送数据包：

```
# ping <host-name-or-IP-address>
```

故障排除步骤

- 验证网线是否插入到主机和交换机。
- 检查链路失败是否只存在于此主机上，或者连接到同一交换机的其它主机上。
- 验证网络电缆和网络接口是否如预期工作。执行硬件诊断步骤，并替换有问题的网线和网络接口卡。
- 如果磁盘中的配置与设备中的配置不匹配，则启动或重启 NetworkManager 会创建一个代表该设备的配置的内存连接。有关详情以及如何避免此问题，请参阅红帽知识库解决方案 [NetworkManager 在重启 NetworkManager 服务后复制连接](#)。

2.5. 使用 NM-CONNECTION-EDITOR 配置以太网连接

如果通过以太网将主机连接到网络，您可以通过 **nm-connection-editor** 应用程序，使用图形界面管理连接的设置。

先决条件

- 服务器配置中存在物理或虚拟以太网网络接口控制器(NIC)。
- 已安装了 GNOME。

流程

1. 打开终端窗口，输入：

```
$ nm-connection-editor
```

2. 选择是否添加新连接配置文件或修改现有连接配置文件：

- 要创建新配置文件：
 - i. 点 + 按钮
 - ii. 选择 **Ethernet** 作为连接类型，然后单击 **Create**。
- 要修改现有配置文件，请双击配置文件条目。

3. 可选：在 **Connection Name** 字段中更新配置文件的名称。

在有多个配置文件的主机上，有意义的名称可以更容易识别配置文件的用途。

4. 如果您创建新配置文件，请在 **Ethernet** 选项卡中选择设备：



5. 根据您的环境，相应地在 **IPv4 Settings** 和 **IPv6 Settings** 选项卡中配置 IP 地址设置：

- 要使用 DHCP 或 IPv6 无状态地址自动配置(SLAAC)，请选择 **Automatic (DHCP)** 作为方法（默认）。
- 要设置静态 IP 地址、网络掩码、默认网关、DNS 服务器和搜索域，请选择 **Manual** 作为方法，并在标签页中填写字段：

<p>Method: Manual</p> <p>Addresses</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <th>Address</th> <th>Netmask</th> <th>Gateway</th> <th>Add</th> <th>Delete</th> </tr> <tr> <td>192.0.2.1</td> <td>24</td> <td>192.0.2.254</td> <td style="background-color: #ADD8E6;">Add</td> <td>Delete</td> </tr> </table> <p>DNS servers: 192.0.2.1</p> <p>Search domains: example.com</p>	Address	Netmask	Gateway	Add	Delete	192.0.2.1	24	192.0.2.254	Add	Delete	<p>Method: Manual</p> <p>Addresses</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <th>Address</th> <th>Prefix</th> <th>Gateway</th> <th>Add</th> <th>Delete</th> </tr> <tr> <td>2001:db8:1::1</td> <td>64</td> <td>2001:db8:1::ffff3</td> <td style="background-color: #ADD8E6;">Add</td> <td>Delete</td> </tr> </table> <p>DNS servers: 2001:db8:1::ffffd</p> <p>Search domains: example.com</p>	Address	Prefix	Gateway	Add	Delete	2001:db8:1::1	64	2001:db8:1::ffff3	Add	Delete
Address	Netmask	Gateway	Add	Delete																	
192.0.2.1	24	192.0.2.254	Add	Delete																	
Address	Prefix	Gateway	Add	Delete																	
2001:db8:1::1	64	2001:db8:1::ffff3	Add	Delete																	

6. 点击 **Save**。

7. 关闭 **nm-connection-editor**。

验证

1. 显示 NIC 的 IP 设置：

```
# ip address show enp1s0
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
group default qlen 1000
    link/ether 52:54:00:17:b8:b6 brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.1/24 brd 192.0.2.255 scope global noprefixroute enp1s0
        valid_lft forever preferred_lft forever
    inet6 2001:db8:1::fffe/64 scope global noprefixroute
        valid_lft forever preferred_lft forever
```

2. 显示 IPv4 默认网关：

```
# ip route show default
default via 192.0.2.254 dev enp1s0 proto static metric 102
```

3. 显示 IPv6 默认网关：

```
# ip -6 route show default
default via 2001:db8:1::ffffe dev enp1s0 proto static metric 102 pref medium
```

4. 显示 DNS 设置：

```
# cat /etc/resolv.conf
search example.com
nameserver 192.0.2.200
nameserver 2001:db8:1::ffbb
```

如果多个连接配置文件同时处于活跃状态，则 **nameserver** 条目的顺序取决于这些配置文件中的 DNS 优先级值和连接类型。

5. 使用 **ping** 工具验证这个主机是否可以向其他主机发送数据包：

```
# ping <host-name-or-IP-address>
```

故障排除步骤

- 验证网线是否插入到主机和交换机。
- 检查链路失败是否只存在于此主机上，或者连接到同一交换机的其它主机上。
- 验证网络电缆和网络接口是否如预期工作。执行硬件诊断步骤，并替换有问题的网线和网络接口卡。
- 如果磁盘中的配置与设备中的配置不匹配，则启动或重启 NetworkManager 会创建一个代表该设备的配置的内存连接。有关详情以及如何避免此问题，请参阅红帽知识库解决方案 [NetworkManager 在重启 NetworkManager 服务后复制连接](#)。

其他资源

- [配置 NetworkManager 以避免使用特定配置集提供默认网关](#)
- [配置 DNS 服务器顺序](#)

2.6. 使用带有接口名称的 **NMSTATECTL** 使用静态 IP 地址配置以太网连接

使用 **nmstatectl** 工具通过 Nmstate API 配置以太网连接。Nmstate API 确保设置配置后结果与配置文件匹配。如果有任何失败，**nmstatectl** 会自动回滚更改以避免系统处于不正确的状态。

您可以使用 Nmstate 配置带有静态 IP 地址、网关和 DNS 设置的以太网连接，并将它们分配给指定的接口名称。

先决条件

- 服务器配置中存在物理或虚拟以太网网络接口控制器(NIC)。
- **nmstate** 软件包已安装。

步骤

1. 创建包含以下内容的 YAML 文件，如 `~/create-ethernet-profile.yml`：

```
---
interfaces:
- name: enp1s0
  type: ethernet
  state: up
  ipv4:
    enabled: true
    address:
    - ip: 192.0.2.1
      prefix-length: 24
    dhcp: false
  ipv6:
    enabled: true
    address:
    - ip: 2001:db8:1::1
      prefix-length: 64
    autoconf: false
    dhcp: false
  routes:
    config:
    - destination: 0.0.0.0/0
      next-hop-address: 192.0.2.254
      next-hop-interface: enp1s0
    - destination: ::/0
      next-hop-address: 2001:db8:1::fffe
      next-hop-interface: enp1s0
  dns-resolver:
    config:
      search:
      - example.com
      server:
      - 192.0.2.200
      - 2001:db8:1::ffbb
```

这些设置使用以下设置为 **enp1s0** 设备定义一个以太网连接配置文件：

- 静态 IPv4 地址 - **192.0.2.1**，子网掩码为 /24

- 静态 IPv6 地址 - **2001:db8:1::1**, 子网掩码为 **/64**
- IPv4 默认网关 - **192.0.2.254**
- IPv6 默认网关 - **2001:db8:1::ffffe**
- IPv4 DNS 服务器 - **192.0.2.200**
- IPv6 DNS 服务器 - **2001:db8:1::ffbb**
- DNS 搜索域 - **example.com**

2. 将设置应用到系统：

```
# nmstatectl apply ~/create-ethernet-profile.yml
```

验证

1. 以 YAML 格式显示当前状态：

```
# nmstatectl show enp1s0
```

2. 显示 NIC 的 IP 设置：

```
# ip address show enp1s0
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
group default qlen 1000
    link/ether 52:54:00:17:b8:b6 brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.1/24 brd 192.0.2.255 scope global noprefixroute enp1s0
        valid_lft forever preferred_lft forever
    inet6 2001:db8:1::ffffe/64 scope global noprefixroute
        valid_lft forever preferred_lft forever
```

3. 显示 IPv4 默认网关：

```
# ip route show default
default via 192.0.2.254 dev enp1s0 proto static metric 102
```

4. 显示 IPv6 默认网关：

```
# ip -6 route show default
default via 2001:db8:1::ffffe dev enp1s0 proto static metric 102 pref medium
```

5. 显示 DNS 设置：

```
# cat /etc/resolv.conf
search example.com
nameserver 192.0.2.200
nameserver 2001:db8:1::ffbb
```

如果多个连接配置文件同时处于活跃状态，则 **nameserver** 条目的顺序取决于这些配置文件中的 DNS 优先级值和连接类型。

- 使用 **ping** 工具验证这个主机是否可以向其他主机发送数据包：

```
# ping <host-name-or-IP-address>
```

其他资源

- 您系统上的 **nmstatectl (8)** 手册页
- /usr/share/doc/nmstate/examples/** 目录

2.7. 使用 NETWORK RHEL 系统角色和接口名称，配置具有静态 IP 地址的以太网连接

要将 Red Hat Enterprise Linux 主机连接到以太网网络，请为网络设备创建一个 NetworkManager 连接配置文件。通过使用 Ansible 和 **network** RHEL 系统角色，您可以自动化此过程，并在 playbook 中定义的主机上远程配置连接配置文件。

您可以使用 **network** RHEL 系统角色配置具有静态 IP 地址、网关和 DNS 的以太网连接，并将它们分配给指定的接口名称。

通常，管理员希望重复使用 playbook，且不会为 Ansible 应该为其分配静态 IP 地址的每个主机维护单独的 playbook。在本例中，您可以在 playbook 中使用变量，并在清单中维护设置。因此，您只需要一个 playbook 就可动态地将单个设置分配给多个主机。

前提条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户对其具有 **sudo** 权限。
- 服务器配置中存在物理或者虚拟以太网设备。
- 受管节点使用 NetworkManager 配置网络。

步骤

- 编辑 **~/inventory** 文件，并将特定于主机的设置附加到主机条目中：

```
managed-node-01.example.com interface=enp1s0 ip_v4=192.0.2.1/24
ip_v6=2001:db8:1::1/64 gateway_v4=192.0.2.254 gateway_v6=2001:db8:1::ffffe

managed-node-02.example.com interface=enp1s0 ip_v4=192.0.2.2/24
ip_v6=2001:db8:1::2/64 gateway_v4=192.0.2.254 gateway_v6=2001:db8:1::ffffe
```

- 创建一个包含以下内容的 playbook 文件，如 **~/playbook.yml**：

```
---
- name: Configure the network
  hosts: managed-node-01.example.com,managed-node-02.example.com
  tasks:
    - name: Ethernet connection profile with static IP address settings
```

```

ansible.builtin.include_role:
  name: redhat.rhel_system_roles.network
vars:
  network_connections:
    - name: "{{ interface }}"
      interface_name: "{{ interface }}"
      type: ethernet
      autoconnect: yes
      ip:
        address:
          - "{{ ip_v4 }}"
          - "{{ ip_v6 }}"
      gateway4: "{{ gateway_v4 }}"
      gateway6: "{{ gateway_v6 }}"
      dns:
        - 192.0.2.200
        - 2001:db8:1::ffbb
      dns_search:
        - example.com
  state: up

```

此 playbook 从清单文件中动态读取每个主机的某些值，并将 playbook 中的静态值用于所有主机的相同的设置。

有关 playbook 中使用的所有变量的详情，请查看控制节点上的 **/usr/share/ansible/roles/rhel-system-roles.network/README.md** 文件。

3. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

4. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

- 查询受管节点的 Ansible 事实，并验证活跃的网络设置：

```
# ansible managed-node-01.example.com -m ansible.builtin.setup
...
"ansible_default_ipv4": {
  "address": "192.0.2.1",
  "alias": "enp1s0",
  "broadcast": "192.0.2.255",
  "gateway": "192.0.2.254",
  "interface": "enp1s0",
  "macaddress": "52:54:00:17:b8:b6",
  "mtu": 1500,
  "netmask": "255.255.255.0",
  "network": "192.0.2.0",
  "prefix": "24",
  "type": "ether"
}
```

```

},
"ansible_default_ipv6": {
    "address": "2001:db8:1::1",
    "gateway": "2001:db8:1::fffe",
    "interface": "enp1s0",
    "macaddress": "52:54:00:17:b8:b6",
    "mtu": 1500,
    "prefix": "64",
    "scope": "global",
    "type": "ether"
},
...
"ansible_dns": {
    "nameservers": [
        "192.0.2.1",
        "2001:db8:1::ffbb"
    ],
    "search": [
        "example.com"
    ]
},
...

```

其他资源

- [/usr/share/ansible/roles/rhel-system-roles.network/README.md](#) 文件
- [/usr/share/doc/rhel-system-roles/network/](#) 目录

2.8. 使用 NETWORK RHEL 系统角色和设备路径，配置具有静态 IP 地址的以太网连接

要将 Red Hat Enterprise Linux 主机连接到以太网网络，请为网络设备创建一个 NetworkManager 连接配置文件。通过使用 Ansible 和 **network** RHEL 系统角色，您可以自动化此过程，并在 playbook 中定义的主机上远程配置连接配置文件。

您可以使用 **network** RHEL 系统角色配置带有静态 IP 地址、网关和 DNS 设置的以太网连接，并根据其路径而不是其名称将它们分配给设备。

前提条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户对其具有 **sudo** 权限。
- 服务器配置中有一个物理或者虚拟以太网设备。
- 受管节点使用 NetworkManager 配置网络。
- 您知道设备的路径。您可以使用 **udevadm info /sys/class/net/<device_name> | grep ID_PATH=** 命令显示设备路径。

步骤

- 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
    - name: Ethernet connection profile with static IP address settings
      ansible.builtin.include_role:
        name: redhat.rhel_system_roles.network
      vars:
        network_connections:
          - name: example
            match:
              path:
                - pci-0000:00:0[1-3].0
                - '&!pci-0000:00:02.0'
            type: ethernet
            autoconnect: yes
            ip:
              address:
                - 192.0.2.1/24
                - 2001:db8:1::1/64
              gateway4: 192.0.2.254
              gateway6: 2001:db8:1::fffe
              dns:
                - 192.0.2.200
                - 2001:db8:1::ffbb
              dns_search:
                - example.com
            state: up
```

示例 playbook 中指定的设置包括以下内容：

match

定义一个应用设置所必须满足的条件。您只能将此变量与 **path** 选项一起使用。

path

定义设备的持久路径。您可以将它设置为固定路径或表达式。其值可以包含修饰符和通配符。这个示例将设置应用到与 PCI ID **0000:00:0[1-3].0** 而不是 **0000:00:02.0** 匹配的设备。

有关 playbook 中使用的所有变量的详情，请查看控制节点上的 `/usr/share/ansible/roles/rhel-system-roles.network/README.md` 文件。

- 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

- 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

- 查询受管节点的 Ansible 事实，并验证活跃的网络设置：

```
# ansible managed-node-01.example.com -m ansible.builtin.setup
...
"ansible_default_ipv4": {
    "address": "192.0.2.1",
    "alias": "enp1s0",
    "broadcast": "192.0.2.255",
    "gateway": "192.0.2.254",
    "interface": "enp1s0",
    "macaddress": "52:54:00:17:b8:b6",
    "mtu": 1500,
    "netmask": "255.255.255.0",
    "network": "192.0.2.0",
    "prefix": "24",
    "type": "ether"
},
"ansible_default_ipv6": {
    "address": "2001:db8:1::1",
    "gateway": "2001:db8:1::fffe",
    "interface": "enp1s0",
    "macaddress": "52:54:00:17:b8:b6",
    "mtu": 1500,
    "prefix": "64",
    "scope": "global",
    "type": "ether"
},
...
"ansible_dns": {
    "nameservers": [
        "192.0.2.1",
        "2001:db8:1::ffbb"
    ],
    "search": [
        "example.com"
    ]
},
...

```

其他资源

- `/usr/share/ansible/roles/rhel-system-roles.network/README.md` 文件
- `/usr/share/doc/rhel-system-roles/network/` 目录

2.9. 使用带有接口名称的 **NMSTATECTL** 使用动态 IP 地址配置以太网连接

使用 **nmstatectl** 工具通过 Nmstate API 配置以太网连接。Nmstate API 确保设置配置后结果与配置文件匹配。如果有任何失败，**nmstatectl** 会自动回滚更改以避免系统处于不正确的状态。

您可以使用 Nmstate 配置以太网连接，从 DHCP 服务器和 IPv6 无状态地址自动配置(SLAAC)检索其 IP 地址、网关和 DNS 设置。您可以将连接配置集分配给指定的接口名称。

先决条件

- 服务器配置中存在物理或虚拟以太网网络接口控制器(NIC)。
- 网络中有 DHCP 服务器。
- nmstate** 软件包已安装。

步骤

- 创建包含以下内容的 YAML 文件，如 `~/create-ethernet-profile.yml`：

```
---
interfaces:
- name: enp1s0
  type: ethernet
  state: up
  ipv4:
    enabled: true
    auto-dns: true
    auto-gateway: true
    auto-routes: true
    dhcp: true
  ipv6:
    enabled: true
    auto-dns: true
    auto-gateway: true
    auto-routes: true
    autoconf: true
    dhcp: true
```

这些设置为 **enp1s0** 设备定义了一个以太网连接配置文件。连接从 DHCP 服务器检索 IPv4 地址、IPv6 地址、默认网关、路由、DNS 服务器和搜索域,以及 IPv6 无状态地址自动配置(SLAAC)。

- 将设置应用到系统：

```
# nmstatectl apply ~/create-ethernet-profile.yml
```

验证

- 以 YAML 格式显示当前状态：

```
# nmstatectl show enp1s0
```

- 显示 NIC 的 IP 设置：

```
# ip address show enp1s0
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
group default qlen 1000
    link/ether 52:54:00:17:b8:b6 brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.1/24 brd 192.0.2.255 scope global noprefixroute enp1s0
        valid_lft forever preferred_lft forever
    inet6 2001:db8:1::ffe/64 scope global noprefixroute
        valid_lft forever preferred_lft forever
```

3. 显示 IPv4 默认网关：

```
# ip route show default
default via 192.0.2.254 dev enp1s0 proto static metric 102
```

4. 显示 IPv6 默认网关：

```
# ip -6 route show default
default via 2001:db8:1::fffe dev enp1s0 proto static metric 102 pref medium
```

5. 显示 DNS 设置：

```
# cat /etc/resolv.conf
search example.com
nameserver 192.0.2.200
nameserver 2001:db8:1::ffbb
```

如果多个连接配置文件同时处于活跃状态，则 **nameserver** 条目的顺序取决于这些配置文件中的 DNS 优先级值和连接类型。

6. 使用 **ping** 程序来验证这个主机是否可以向其它主机发送数据包：

```
# ping <host-name-or-IP-address>
```

其他资源

- 您系统上的 **nmstatectl (8)** 手册页
- **/usr/share/doc/nmstate/examples/** 目录

2.10. 使用 NETWORK RHEL 系统角色和接口名称，配置具有动态 IP 地址的以太网连接

要将 Red Hat Enterprise Linux 主机连接到以太网网络，请为网络设备创建一个 NetworkManager 连接配置文件。通过使用 Ansible 和 **network** RHEL 系统角色，您可以自动化此过程，并在 playbook 中定义的主机上远程配置连接配置文件。

您可以使用 **network** RHEL 系统角色配置以太网连接，该连接从 DHCP 服务器检索其 IP 地址、网关和 DNS 设置，以及 IPv6 无状态地址自动配置(SLAAC)。使用此角色，您可以将连接配置文件分配给指定的接口名称。

先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户对其具有 **sudo** 权限。
- 服务器配置中存在物理或者虚拟以太网设备。
- 网络中有 DHCP 服务器和 SLAAC。

- 受管节点使用 NetworkManager 服务来配置网络。

步骤

- 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
    - name: Ethernet connection profile with dynamic IP address settings
      ansible.builtin.include_role:
        name: redhat.rhel_system_roles.network
      vars:
        network_connections:
          - name: enp1s0
            interface_name: enp1s0
            type: ethernet
            autoconnect: yes
            ip:
              dhcp4: yes
              auto6: yes
            state: up
```

示例 playbook 中指定的设置包括以下内容：

dhcp4: yes

启用来自 DHCP、PPP 或类似服务的自动 IPv4 地址分配。

auto6: yes

启用 IPv6 自动配置。默认情况下，NetworkManager 使用路由器通告。如果路由器宣布 **managed** 标记，则 NetworkManager 会从 DHCPv6 服务器请求 IPv6 地址和前缀。

有关 playbook 中使用的所有变量的详情，请查看控制节点上的 `/usr/share/ansible/roles/rhel-system-roles.network/README.md` 文件。

- 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

- 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

- 查询受管节点的 Ansible 事实，并验证接口是否收到 IP 地址和 DNS 设置：

```
# ansible managed-node-01.example.com -m ansible.builtin.setup
...
  "ansible_default_ipv4": {
    "address": "192.0.2.1",
```

```

"alias": "enp1s0",
"broadcast": "192.0.2.255",
"gateway": "192.0.2.254",
"interface": "enp1s0",
"macaddress": "52:54:00:17:b8:b6",
"mtu": 1500,
"netmask": "255.255.255.0",
"network": "192.0.2.0",
"prefix": "24",
"type": "ether"
},
"ansible_default_ipv6": {
"address": "2001:db8:1::1",
"gateway": "2001:db8:1::fffe",
"interface": "enp1s0",
"macaddress": "52:54:00:17:b8:b6",
"mtu": 1500,
"prefix": "64",
"scope": "global",
"type": "ether"
},
...
"ansible_dns": {
"nameservers": [
"192.0.2.1",
"2001:db8:1::ffbb"
],
"search": [
"example.com"
]
},
...
...

```

其他资源

- [/usr/share/ansible/roles/rhel-system-roles.network/README.md](#) 文件
- [/usr/share/doc/rhel-system-roles/network/](#) 目录

2.11. 使用 NETWORK RHEL 系统角色和设备路径，配置具有动态 IP 地址的以太网连接

要将 Red Hat Enterprise Linux 主机连接到以太网网络，请为网络设备创建一个 NetworkManager 连接配置文件。通过使用 Ansible 和 **network** RHEL 系统角色，您可以自动化此过程，并在 playbook 中定义的主机上远程配置连接配置文件。

您可以使用 **network** RHEL 系统角色配置以太网连接，该连接从 DHCP 服务器检索其 IP 地址、网关和 DNS 设置，以及 IPv6 无状态地址自动配置(SLAAC)。角色可以根据其路径而不是接口名称将连接配置文件分配给设备。

先决条件

- 您已准备好控制节点和受管节点

- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户对其具有 **sudo** 权限。
- 服务器配置中有一个物理或者虚拟以太网设备。
- 网络中有 DHCP 服务器和 SLAAC。
- 受管主机使用 NetworkManager 配置网络。
- 您知道设备的路径。您可以使用 **udevadm info /sys/class/net/<device_name> | grep ID_PATH=** 命令显示设备路径。

步骤

1. 创建一个包含以下内容的 playbook 文件，如 **~/playbook.yml**：

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
    - name: Ethernet connection profile with dynamic IP address settings
      ansible.builtin.include_role:
        name: redhat.rhel_system_roles.network
      vars:
        network_connections:
          - name: example
            match:
              path:
                - pci-0000:00:0[1-3].0
                - '&!pci-0000:00:02.0'
              type: ethernet
              autoconnect: yes
              ip:
                dhcp4: yes
                auto6: yes
              state: up
```

示例 playbook 中指定的设置包括以下内容：

match: path

定义一个应用设置所必须满足的条件。您只能将此变量与 **path** 选项一起使用。

path: <path_and_expressions>

定义设备的持久路径。您可以将它设置为固定路径或表达式。其值可以包含修饰符和通配符。这个示例将设置应用到与 PCI ID **0000:00:0[1-3].0** 而不是 **0000:00:02.0** 匹配的设备。

dhcp4: yes

启用来自 DHCP、PPP 或类似服务的自动 IPv4 地址分配。

auto6: yes

启用 IPv6 自动配置。默认情况下，NetworkManager 使用路由器通告。如果路由器宣布 **managed** 标记，则 NetworkManager 会从 DHCPv6 服务器请求 IPv6 地址和前缀。

有关 playbook 中使用的所有变量的详情，请查看控制节点上的 **/usr/share/ansible/roles/rhel-system-roles.network/README.md** 文件。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

- 查询受管节点的 Ansible 事实，并验证接口是否收到 IP 地址和 DNS 设置：

```
# ansible managed-node-01.example.com -m ansible.builtin.setup
...
"ansible_default_ipv4": {
    "address": "192.0.2.1",
    "alias": "enp1s0",
    "broadcast": "192.0.2.255",
    "gateway": "192.0.2.254",
    "interface": "enp1s0",
    "macaddress": "52:54:00:17:b8:b6",
    "mtu": 1500,
    "netmask": "255.255.255.0",
    "network": "192.0.2.0",
    "prefix": "24",
    "type": "ether"
},
"ansible_default_ipv6": {
    "address": "2001:db8:1::1",
    "gateway": "2001:db8:1::fffe",
    "interface": "enp1s0",
    "macaddress": "52:54:00:17:b8:b6",
    "mtu": 1500,
    "prefix": "64",
    "scope": "global",
    "type": "ether"
},
...
"ansible_dns": {
    "nameservers": [
        "192.0.2.1",
        "2001:db8:1::ffbb"
    ],
    "search": [
        "example.com"
    ]
},
...

```

其他资源

- [/usr/share/ansible/roles/rhel-system-roles.network/README.md](#) 文件

- /usr/share/doc/rhel-system-roles/network/ 目录

2.12. 按接口名称，使用单个连接配置文件配置多个以太网接口

在大多数情况下，一个连接配置文件包含一个网络设备的设置。但是，当您在连接配置文件中设置接口名称时，NetworkManager 也支持通配符。如果主机在具有动态 IP 地址分配的以太网之间漫游，则您可以使用此功能创建可用于多个以太网接口的单一连接配置文件。

先决条件

- 服务器配置中存在多个物理或虚拟以太网设备。
- 网络中有 DHCP 服务器。
- 主机上不存在连接配置文件。

步骤

1. 添加可应用于以 **enp** 开头的所有接口名称的连接配置文件：

```
# nmcli connection add con-name "Wired connection 1" connection.multi-connect
multiple match.interface-name enp* type ethernet
```

验证

1. 显示单个连接配置文件的所有设置：

```
# nmcli connection show "Wired connection 1"
connection.id:           Wired connection 1
...
connection.multi-connect:   3 (multiple)
match.interface-name:     enp*
...
...
```

3 表示可以在特定时间被多次激活的接口。连接配置文件使用与 **match.interface-name** 参数中的模式匹配的所有设备，因此连接配置文件具有相同的通用唯一识别符(UUID)。

2. 显示连接的状态：

```
# nmcli connection show
NAME          UUID              TYPE      DEVICE
...
Wired connection 1  6f22402e-c0cc-49cf-b702-eaf0cd5ea7d1  ethernet  enp7s0
Wired connection 1  6f22402e-c0cc-49cf-b702-eaf0cd5ea7d1  ethernet  enp8s0
Wired connection 1  6f22402e-c0cc-49cf-b702-eaf0cd5ea7d1  ethernet  enp9s0
```

其他资源

- 您系统上的 **nmcli (1)** 手册页
- **nm-settings(5)** 手册页

2.13. 使用 PCI ID 为多个以太网接口配置一个连接配置文件

PCI ID 是连接到系统的设备的唯一标识符。连接配置文件根据 PCI ID 列表按匹配的接口来添加多个设备。您可以使用这个流程将多个设备 PCI ID 连接到一个连接配置文件。

先决条件

- 服务器配置中存在多个物理或虚拟以太网设备。
- 网络中有 DHCP 服务器。
- 主机上不存在连接配置文件。

步骤

1. 识别设备路径。例如，要显示以 `enp` 开头的所有接口的设备路径，请输入：

```
# udevadm info /sys/class/net/enp* | grep ID_PATH=
...
E: ID_PATH=pci-0000:07:00.0
E: ID_PATH=pci-0000:08:00.0
```

2. 添加可应用于匹配 `0000:00:0[7-8].0` 表达式的所有 PCI ID 的连接配置文件：

```
# nmcli connection add type ethernet connection.multi-connect multiple match.path
"pci-0000:07:00.0 pci-0000:08:00.0" con-name "Wired connection 1"
```

验证

1. 显示连接的状态：

```
# nmcli connection show
NAME           UUID             TYPE      DEVICE
Wired connection 1  9cee0958-512f-4203-9d3d-b57af1d88466  ethernet  enp7s0
Wired connection 1  9cee0958-512f-4203-9d3d-b57af1d88466  ethernet  enp8s0
...
```

2. 显示连接配置集的所有设置：

```
# nmcli connection show "Wired connection 1"
connection.id:          Wired connection 1
...
connection.multi-connect: 3 (multiple)
match.path:            pci-0000:07:00.0,pci-0000:08:00.0
...
```

此连接配置文件使用 PCI ID 与 `match.path` 参数中的模式匹配的所有设备，因此连接配置文件具有相同的全局唯一标识符(UUID)。

其他资源

- 您系统上的 `nmcli (1)` 手册页
- `nm-settings(5)` 手册页

第 3 章 配置网络绑定

网络绑定是一种组合或聚合物理和虚拟网络接口的方法，以提供高吞吐量或冗余的逻辑接口。在绑定中，内核只处理所有操作。您可以在不同类型的设备中创建绑定，如以太网设备或 VLAN。

Red Hat Enterprise Linux 为管理员提供不同的选项来配置团队设备。例如：

- 使用 **nmcli** 使用命令行配置绑定连接。
- 通过 RHEL web 控制台使用 Web 浏览器配置绑定连接。
- 使用 **nmtui** 在基于文本的用户界面中配置绑定连接。
- 使用 **nm-connection-editor** 应用程序在图形界面中配置绑定连接。
- 使用 **nmstatectl** 通过 Nmstate API 配置绑定连接。
- 使用 RHEL 系统角色在一个或多个主机上自动化绑定配置。

3.1. 了解控制器和端口接口的默认行为

在使用 **NetworkManager** 服务管理或排除团队或绑定端口接口故障时，请考虑以下默认行为：

- 启动控制器接口不会自动启动端口接口。
- 启动端口接口总会启动控制器接口。
- 停止控制器接口也会停止端口接口。
- 没有端口的控制器可以启动静态 IP 连接。
- 没有端口的控制器在启动 DHCP 连接时会等待端口。
- 当您添加具有载体的端口时，等待端口且具有 DHCP 连接的控制器会完成。
- 当您添加没有载体的端口时，具有等待端口的 DHCP 连接的控制器继续等待。

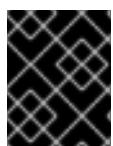
3.2. 依赖绑定模式的上游交换机配置

根据您要使用的绑定模式，您必须在交换机上配置端口：

绑定模式	交换机上的配置
0 - balance-rr	需要启用静态 EtherChannel，而不是链路聚合控制协议(LACP)协商。
1 - active-backup	交换机上不需要任何配置。
2 - balance-xor	需要启用静态 EtherChannel，而不是 LACP 协商。
3 - broadcast	需要启用静态 EtherChannel，而不是 LACP 协商。

绑定模式	交换机上的配置
4 - 802.3ad	需要启用 LACP 协商的 EtherChannel。
5 - balance-tlb	交换机上不需要任何配置。
6 - balance-alb	交换机上不需要任何配置。

有关如何配置交换机的详情，请查看交换机的文档。



重要

某些网络绑定的功能，比如故障切换机制，不支持不通过网络交换机的直接电缆连接。详情请查看红帽知识库解决方案 [使用交叉网线的直接连接支持绑定吗](#)。

3.3. 使用 nmcli 配置网络绑定

要在命令行中配置网络绑定，请使用 **nmcli** 工具。

先决条件

- 在服务器中安装两个或者两个以上物理或者虚拟网络设备。
- 要将以太网设备用作绑定的端口，必须在服务器中安装物理或者虚拟以太网设备。
- 要使用 team、bridge 或 VLAN 设备作为绑定的端口，您可以在创建绑定时创建这些设备，或者预先创建它们，如：
 - 使用 nmcli 配置网络团队
 - 使用 nmcli 配置网桥
 - 使用 nmcli 配置 VLAN 标记

步骤

- 创建绑定接口：

```
# nmcli connection add type bond con-name bond0 ifname bond0 bond.options
"mode=active-backup"
```

这个命令会创建一个使用 **active-backup** 模式、名为 **bond0** 的绑定。

要额外设置介质独立接口(MII)监控间隔，请在 **bond.options** 属性中添加 **miimon=interval** 选项，例如：

```
# nmcli connection add type bond con-name bond0 ifname bond0 bond.options
"mode=active-backup,miimon=1000"
```

- 显示网络接口以及您要添加到绑定中的接口名称：

```
# nmcli device status
DEVICE  TYPE      STATE      CONNECTION
enp7s0  ethernet  disconnected --
enp8s0  ethernet  disconnected --
bridge0 bridge    connected   bridge0
bridge1 bridge    connected   bridge1
...
```

在本例中：

- 没有配置 **enp7s0** 和 **enp8s0**。要将这些设备用作端口，请在下一步中添加连接配置集。
- **bridge0** 和 **bridge1** 都有现有的连接配置文件。要将这些设备用作端口，请在下一步中修改其配置集。

3. 为绑定分配接口：

- a. 如果没有配置您要分配给绑定的接口，为其创建新的连接配置集：

```
# nmcli connection add type ethernet slave-type bond con-name bond0-port1
ifname enp7s0 master bond0
# nmcli connection add type ethernet slave-type bond con-name bond0-port2
ifname enp8s0 master bond0
```

这些命令为 **enp7s0** 和 **enp8s0** 创建配置文件，并将它们添加到 **bond0** 连接中。

- b. 为绑定分配现有连接配置集：

- i. 将这些连接的 **master** 参数设置为 **bond0**：

```
# nmcli connection modify bridge0 master bond0
# nmcli connection modify bridge1 master bond0
```

这些命令将名为 **bridge0** 和 **bridge1** 的现有连接配置文件分配给 **bond0** 连接。

- ii. 重新激活连接：

```
# nmcli connection up bridge0
# nmcli connection up bridge1
```

4. 配置 IPv4 设置：

- 要为 **bond0** 连接设置静态 IPv4 地址、网络掩码、默认网关和 DNS 服务器设置，请输入：

```
# nmcli connection modify bond0 ipv4.addresses '192.0.2.1/24' ipv4.gateway
'192.0.2.254' ipv4.dns '192.0.2.253' ipv4.dns-search 'example.com' ipv4.method
manual
```

- 要使用 DHCP，不需要任何操作。
- 如果您计划将此绑定设备用作其它设备的端口，则不需要任何操作。

5. 配置 IPv6 设置：

- 要为 **bond0** 连接设置静态 IPv6 地址、网络掩码、默认网关和 DNS 服务器设置，请输入：

```
# nmcli connection modify bond0 ipv6.addresses '2001:db8:1::1/64' ipv6.gateway
'2001:db8:1::ffffe' ipv6.dns '2001:db8:1::ffffd' ipv6.dns-search 'example.com'
ipv6.method manual
```

- 要使用无状态地址自动配置(SLAAC)，不需要采取任何操作。
 - 如果您计划将此绑定设备用作其它设备的端口，则不需要任何操作。
6. 可选：如果要在绑定端口上设置任何参数，请使用以下命令：

```
# nmcli connection modify bond0-port1 bond-port.<parameter> <value>
```

7. 激活连接：

```
# nmcli connection up bond0
```

8. 验证端口是否已连接，并且 **CONNECTION** 列是否显示端口的连接名称：

```
# nmcli device
DEVICE  TYPE      STATE      CONNECTION
...
enp7s0  ethernet  connected  bond0-port1
enp8s0  ethernet  connected  bond0-port2
```

当您激活连接的任何端口时，NetworkManager 也激活绑定，但不会激活它的其它端口。您可以配置 Red Hat Enterprise Linux 在启用绑定时自动启用所有端口：

- a. 启用绑定连接的 **connection.autoconnect-slaves** 参数：

```
# nmcli connection modify bond0 connection.autoconnect-slaves 1
```

- b. 重新激活桥接：

```
# nmcli connection up bond0
```

验证

1. 从其中一个网络设备中临时拔掉网线，并检查绑定中的其他设备是否在处理流量。
请注意，无法使用软件工具正确测试链路失败事件。停用连接的工具（如 **nmcli**），只显示绑定驱动程序可以处理端口配置的更改，而不是实际的链接失败事件。
2. 显示绑定状态：

```
# cat /proc/net/bonding/bond0
```

3.4. 使用 RHEL WEB 控制台配置网络绑定

如果您希望使用基于 Web 浏览器的界面管理网络设置，请使用 RHEL web 控制台配置网络绑定。

先决条件

- 在服务器中安装两个或者两个以上物理或者虚拟网络设备。

- 要将以太网设备用作绑定的成员，必须在服务器中安装物理或者虚拟以太网设备。
- 要将 team、bridge 或 VLAN 设备用作绑定成员，请预先创建它们，如：
 - 使用 RHEL web 控制台配置网络团队
 - 使用 RHEL web 控制台配置网桥
 - 使用 RHEL web 控制台配置 VLAN 标记
- 您已安装了 RHEL 8 web 控制台。
- 您已启用了 cockpit 服务。
- 您的用户帐户被允许登录到 web 控制台。
具体步骤请参阅[安装并启用 Web 控制台](#)。

流程

1. 登录到 RHEL 8 web 控制台。
详情请参阅[登录到 web 控制台](#)。
2. 在屏幕左侧的导航中选择 **Networking** 选项卡。
3. 在 **Interfaces** 部分点 **Add bond**。
4. 输入您要创建的绑定设备名称。
5. 选择应该是绑定成员的接口。
6. 选择绑定模式。
如果您选择 **Active backup**，Web 控制台会显示额外的 **Primary** 字段，您可以在其中选择首选的活动设备。
7. 设置链路监控模式。例如，当您使用 **Adaptive 负载均衡** 模式时，将它设置为 **ARP**。
8. 可选：调整监控间隔、链接延迟和连接延迟设置。通常，您只需要更改默认值以进行故障排除。

Bond settings

Name	bond0
Interfaces	<input checked="" type="checkbox"/> enp7s0 <input checked="" type="checkbox"/> enp8s0
MAC	
Mode	Active backup
Primary	enp7s0
Link monitoring	MII (recommended)
Monitoring interval	100
Link up delay	0
Link down delay	0
<input type="button" value="Apply"/> <input type="button" value="Cancel"/>	

9. 点应用。

10. 默认情况下，绑定使用动态 IP 地址。如果要设置静态 IP 地址：

- a. 在 Interfaces 部分点绑定的名称。
- b. 点您要配置的协议旁的 Edit。
- c. 选择 Addresses 旁的 Manual，并输入 IP 地址、前缀和默认网关。
- d. 在 DNS 部分，点 + 按钮，并输入 DNS 服务器的 IP 地址。重复此步骤来设置多个 DNS 服务器。
- e. 在 DNS search domains 部分中，点 + 按钮并输入搜索域。

- f. 如果接口需要静态路由, 请在 **Routes** 部分配置它们。

The screenshot shows the 'IPv4 settings' dialog in NMTUI. At the top, there's a dropdown menu set to 'Manual' and a '+' button. Below it, there's a table for static IP configuration:

Address	Prefix length or netmask	Gateway
192.0.2.1	24	192.0.2.254

Below the table, there's a 'DNS' section with a toggle switch set to 'Automatic' and a '+' button. It includes a 'Server' field containing '192.0.2.253' and a '-' button.

Further down is a 'DNS search domains' section with a toggle switch set to 'Automatic' and a '+' button. It includes a 'Search domain' field containing 'example.com' and a '-' button.

At the bottom of the dialog are 'Apply' and 'Cancel' buttons.

- g. 点 应用

验证

1. 在屏幕左侧的导航中选择 **Networking** 选项卡, 并检查接口上是否有传入和传出流量 :

The screenshot shows the 'Interfaces' list in NMTUI. It displays a table of network interfaces with columns for Name, IP address, Sending, and Receiving traffic rates. The interface 'bond0' is listed with an IP address of 192.0.2.1/24, a sending rate of 1.11 Mbps, and a receiving rate of 61.2 Mbps.

Name	IP address	Sending	Receiving
bond0	192.0.2.1/24	1.11 Mbps	61.2 Mbps

2. 从其中一个网络设备中临时删除网线, 并检查绑定中的其他设备是否在处理流量。
请注意, 无法使用软件工具正确测试链路失败事件。取消激活连接的工具 (如 Web 控制台) 只显示处理成员配置更改且没有实际链路失败事件的能力。
3. 显示绑定状态 :

```
# cat /proc/net/bonding/bond0
```

3.5. 使用 NMTUI 配置网络绑定

nmtui 应用程序为 NetworkManager 提供了一个基于文本的用户界面。您可以使用 **nmtui** 在没有图形界面的主机上配置网络绑定。



注意

在 **nmtui** 中：

- 使用光标键导航。
- 选择一个按钮并按 **Enter** 键。
- 使用 **空格** 选择和清除复选框。
- 要返回上一个屏幕，请使用 **ESC**。

先决条件

- 在服务器中安装两个或者两个以上物理或者虚拟网络设备。
- 要将以太网设备用作绑定的端口，必须在服务器中安装物理或者虚拟以太网设备。

流程

1. 如果您不知道您要在其上配置网络绑定的网络设备名称，请显示可用的设备：

```
# nmcli device status
DEVICE  TYPE      STATE      CONNECTION
enp7s0   ethernet  unavailable --
enp8s0   ethernet  unavailable --
...
```

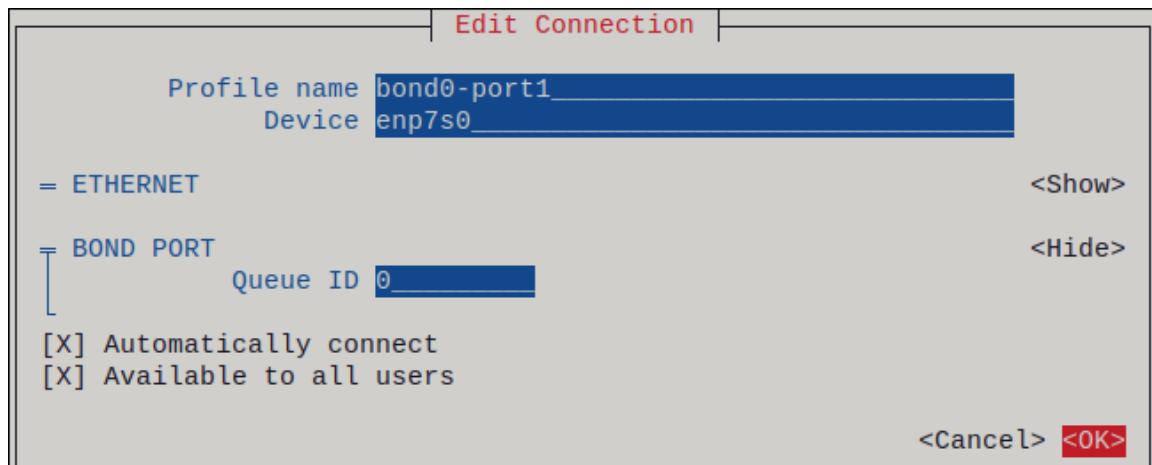
2. 启动 **nmtui**：

```
# nmtui
```

3. 选择 **Edit a connection**，然后按 **Enter**。
4. 按 **Add**。
5. 从网络类型列表中选择 **Bond**，然后按 **Enter** 键。
6. 可选：为要创建的 NetworkManager 配置集输入一个名称。
在有多个配置文件的主机上，有意义的名称可以更容易识别配置文件的用途。
7. 在 **Device** 字段中输入要创建的绑定设备名称。
8. 为要创建的绑定添加端口：
 - a. 按 **Slaves** 列表旁边的 **Add**。
 - b. 选择您要添加为绑定的端口的接口的类型，例如 **Ethernet**。
 - c. 可选：为这个绑定端口输入要创建的 NetworkManager 配置文件的名称。
 - d. 在 **Device** 字段中输入端口的设备名称。

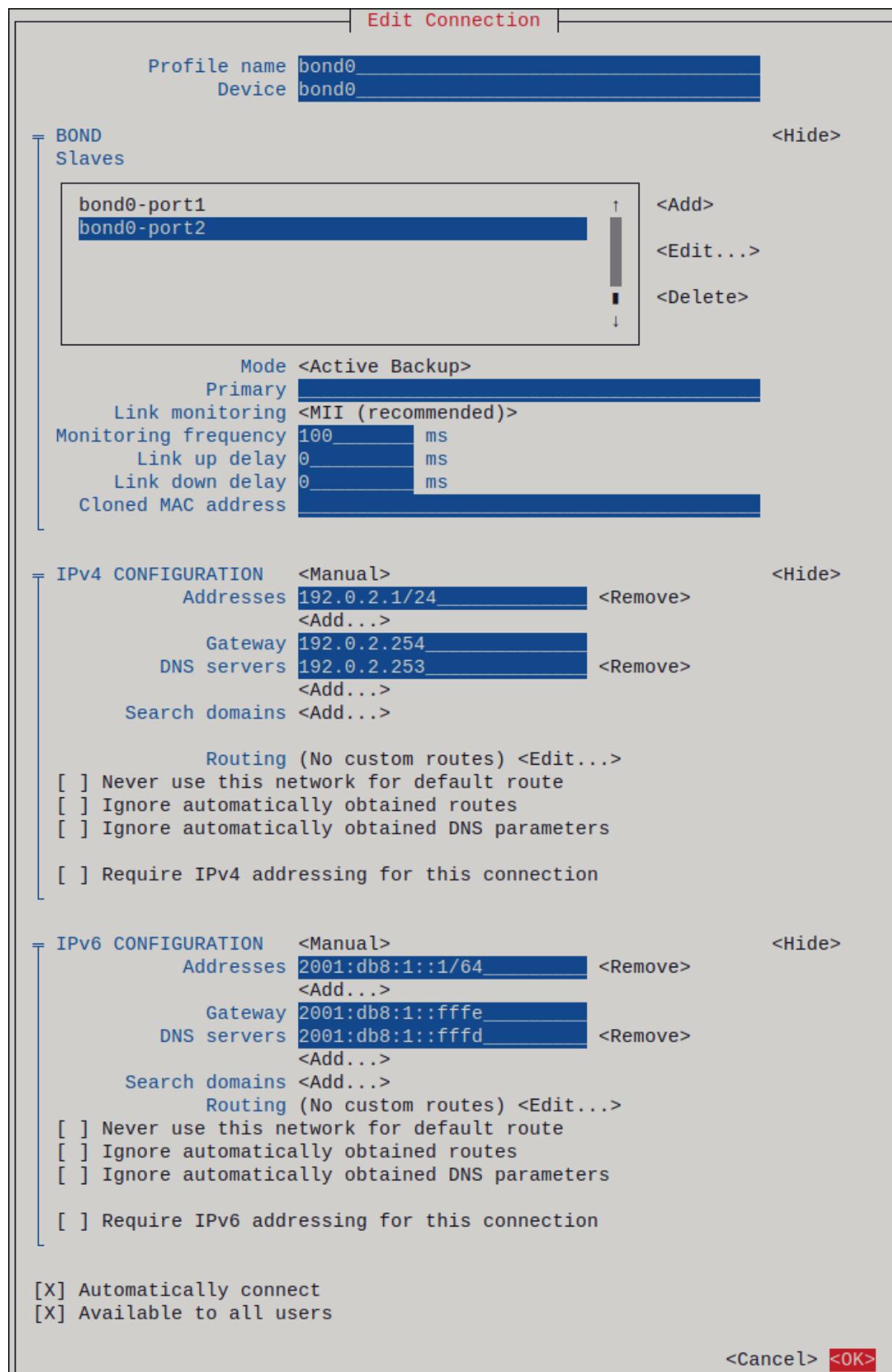
- e. 按 **OK** 返回到绑定设置窗口。

图 3.1. 将以太网设备作为端口添加到绑定



- f. 重复这些步骤，来向绑定添加更多的端口。
9. 设置绑定模式。根据您设置的值，**nmtui** 会显示与所选模式相关的设置的额外字段。
 10. 根据您的环境，相应地在 **IPv4 configuration** 和 **IPv6 configuration** 配置区域中配置 IP 地址。为此，请按这些区域旁边的按钮，并选择：
 - **Disabled**，如果绑定不需要 IP 地址。
 - **Automatic**，如果 DHCP 服务器或无状态地址自动配置(SLAAC)动态将 IP 地址分配给绑定。
 - **Manual**，如果网络需要静态 IP 地址设置。在这种情况下，您必须填写更多字段：
 - i. 在您要配置的协议旁边按 **Show** 以显示其他字段。
 - ii. 按 **Addresses** 旁边的 **Add**，并输入无类别域间路由(CIDR)格式的 IP 地址和子网掩码。如果没有指定子网掩码，NetworkManager 会为 IPv4 地址设置 /32 子网掩码，并为 IPv6 地址设置 /64。
 - iii. 输入默认网关的地址。
 - iv. 按 **DNS servers** 旁边的 **Add**，并输入 DNS 服务器地址。
 - v. 按 **Search domains** 旁边的 **Add**，并输入 DNS 搜索域。

图 3.2. 具有静态 IP 地址设置的绑定连接的示例



11. 按 **OK** 创建并自动激活新连接。

12. 按 **Back** 返回到主菜单。
13. 选择 **Quit**, 然后按 **Enter** 键关闭 **nmtui** 应用程序。

验证

1. 从其中一个网络设备临时拔掉网线，并检查绑定中的其他设备是否在处理流量。
请注意，无法使用软件工具正确测试链路失败事件。停用连接的工具（如 **nmcli**），只显示绑定驱动程序可以处理端口配置的更改，而不是实际的链接失败事件。
2. 显示绑定状态：

```
# cat /proc/net/bonding/bond0
```

3.6. 使用 NM-CONNECTION-EDITOR 配置网络绑定

如果使用带有图形界面的 Red Hat Enterprise Linux，您可以使用 **nm-connection-editor** 应用程序配置网络绑定。

请注意：**nm-connection-editor** 只能向绑定添加新端口。要使用现有连接配置文件作为端口，请使用 **nmcli** 工具创建绑定，[如使用 nmcli 配置网络绑定](#) 中所述。

先决条件

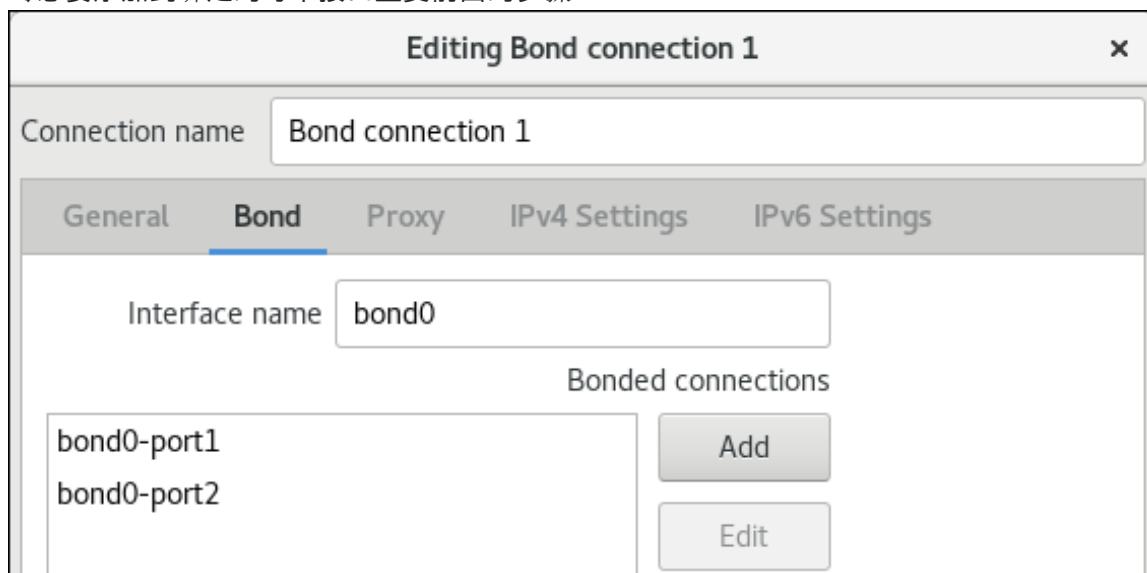
- 在服务器中安装两个或者两个以上物理或者虚拟网络设备。
- 要将以太网设备用作绑定的端口，必须在服务器中安装物理或者虚拟以太网设备。
- 要使用 team、bond 或 VLAN 设备作为绑定的端口，请确保这些设备还没有配置。

流程

1. 打开一个终端，输入 **nm-connection-editor**：
- ```
$ nm-connection-editor
```
2. 点击 **+** 按钮来添加一个新的连接。
  3. 选择 **Bond** 连接类型，然后单击 **Create**。
  4. 在 **Bond** 选项卡中：
    - a. 可选：在 **Interface name** 字段中设置绑定接口的名称。
    - b. 点 **Add** 按钮将网络接口作为端口添加到绑定。
      - i. 选择接口的连接类型。例如，为有线连接选择 **Ethernet**。
      - ii. 可选：为端口设置连接名称。
      - iii. 如果您为以太网设备创建一个连接配置文件，请打开 **Ethernet** 选项卡，在 **Device** 字段中选择您要作为端口添加到绑定的网络接口。如果您选择了不同的设备类型，请相应地进行配置。请注意，您只能在没有配置的绑定中使用以太网接口。

iv. 点 **Save**。

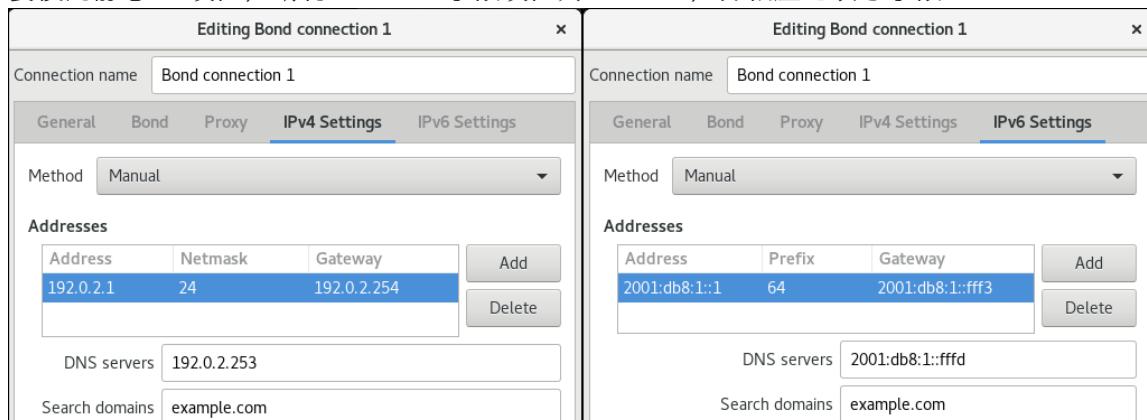
c. 对您要添加到绑定的每个接口重复前面的步骤：



d. 可选：设置其他选项，如介质独立接口（MII）监控间隔。

5. 在 **IPv4 Settings** 和 **IPv6 Settings** 标签页中配置 IP 地址设置：

- 如果您计划将此网桥设备用作其它设备的端口，请将 **Method** 字段设置为 **Disabled**。
- 要使用 DHCP，请将 **Method** 字段保留为默认值 **Automatic (DHCP)**。
- 要使用静态 IP 设置，请将 **Method** 字段设置为 **Manual**，并相应地填写字段：



6. 点击 **Save**。

7. 关闭 **nm-connection-editor**。

## 验证

- 从其中一个网络设备临时拔掉网线，并检查绑定中的其他设备是否在处理流量。  
请注意，无法使用软件工具正确测试链路失败事件。停用连接的工具（如 **nmcli**），只显示绑定驱动程序可以处理端口配置的更改，而不是实际的链接失败事件。
- 显示绑定状态：

```
cat /proc/net/bonding/bond0
```

## 其他资源

- 配置 NetworkManager 以避免使用特定配置集提供默认网关
- 使用 nm-connection-editor 配置网络团队
- 使用 nm-connection-editor 配置网桥
- 使用 nm-connection-editor 配置 VLAN 标记

## 3.7. 使用 NMSTATECTL 配置网络绑定

使用 **nmstatectl** 工具通过 Nmstate API 配置网络绑定。Nmstate API 确保设置配置后结果与配置文件匹配。如果有任何失败，**nmstatectl** 会自动回滚更改以避免系统处于不正确的状态。

根据您的环境，相应地调整 YAML 文件。例如，要使用与绑定中以太网适配器不同的设备，请调整您在绑定中使用的端口的 **base-iface** 属性和 **type** 属性。

### 先决条件

- 在服务器中安装两个或者两个以上物理或者虚拟网络设备。
- 要将以太网设备用作绑定中的端口，必须在服务器中安装物理或者虚拟以太网设备。
- 要在绑定中使用团队、网桥或 VLAN 设备作为端口，请在 **port** 列表中设置接口名称，并定义相应的接口。
- **nmstate** 软件包已安装。

### 流程

1. 创建一个包含以下内容的 YAML 文件，如 `~/create-bond.yml`：

```

interfaces:
- name: bond0
 type: bond
 state: up
 ipv4:
 enabled: true
 address:
 - ip: 192.0.2.1
 prefix-length: 24
 dhcp: false
 ipv6:
 enabled: true
 address:
 - ip: 2001:db8:1::1
 prefix-length: 64
 autoconf: false
 dhcp: false
 link-aggregation:
 mode: active-backup
 port:
 - enp1s0
```

```

 - enp7s0
- name: enp1s0
 type: ethernet
 state: up
- name: enp7s0
 type: ethernet
 state: up

routes:
 config:
 - destination: 0.0.0.0/0
 next-hop-address: 192.0.2.254
 next-hop-interface: bond0
 metric: 300
 - destination: ::/0
 next-hop-address: 2001:db8:1::fffe
 next-hop-interface: bond0
 metric: 300

dns-resolver:
 config:
 search:
 - example.com
 server:
 - 192.0.2.200
 - 2001:db8:1::ffbb

```

这些设置使用以下设置定义网络绑定：

- 绑定中的网络接口：**enp1s0** 和 **enp7s0**
- 模式：**active-backup**
- 静态 IPv4 地址：**192.0.2.1**, 子网掩码为 /**24**
- 静态 IPv6 地址：**2001:db8:1::1**子网掩码为 /**64**
- IPv4 默认网关：**192.0.2.254**
- IPv6 默认网关：**2001:db8:1::fffe**
- IPv4 DNS 服务器：**192.0.2.200**
- IPv6 DNS 服务器：**2001:db8:1::ffbb**
- DNS 搜索域：**example.com**

2. 将设置应用到系统：

```
nmstatectl apply ~/create-bond.yml
```

验证

1. 显示设备和连接的状态：

```
nmcli device status
DEVICE TYPE STATE CONNECTION
bond0 bond connected bond0
```

2. 显示连接配置集的所有设置：

```
nmcli connection show bond0
connection.id: bond0
connection.uuid: 79cbc3bd-302e-4b1f-ad89-f12533b818ee
connection.stable-id: --
connection.type: bond
connection.interface-name: bond0
...
...
```

3. 以 YAML 格式显示连接设置：

```
nmstatectl show bond0
```

## 其他资源

- 您系统上的 **nmstatectl (8)** 手册页
- **/usr/share/doc/nmstate/examples/** 目录

## 3.8. 使用 NETWORK RHEL 系统角色配置网络绑定

您可以将网络接口组合在一个绑定中，以提供具有高吞吐量或冗余的逻辑接口。要配置绑定，请创建一个 NetworkManager 连接配置文件。通过使用 Ansible 和 **network** RHEL 系统角色，您可以自动化此过程，并在 playbook 中定义的主机上远程配置连接配置文件。

您可以使用 **network** RHEL 系统角色配置网络绑定，如果用于绑定的父设备的连接配置文件不存在，角色也可以创建它。

### 先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 您用于连接到受管节点的帐户对它们具有 **sudo** 权限。
- 在服务器中安装两个或者两个以上物理或者虚拟网络设备。

### 流程

1. 创建一个包含以下内容的 playbook 文件，如 **~/playbook.yml**：

```

- name: Configure the network
 hosts: managed-node-01.example.com
 tasks:
 - name: Bond connection profile with two Ethernet ports
 ansible.builtin.include_role:
```

```

name: redhat.rhel_system_roles.network
vars:
 network_connections:
 # Bond profile
 - name: bond0
 type: bond
 interface_name: bond0
 ip:
 dhcp4: yes
 auto6: yes
 bond:
 mode: active-backup
 state: up

 # Port profile for the 1st Ethernet device
 - name: bond0-port1
 interface_name: enp7s0
 type: ethernet
 controller: bond0
 state: up

 # Port profile for the 2nd Ethernet device
 - name: bond0-port2
 interface_name: enp8s0
 type: ethernet
 controller: bond0
 state: up

```

示例 playbook 中指定的设置包括如下：

#### **type: <profile\_type>**

设置要创建的配置文件的类型。示例 playbook 创建三个连接配置文件：一个用于绑定，两个用于以太网设备。

#### **dhcp4: yes**

启用来自 DHCP、PPP 或类似服务的自动 IPv4 地址分配。

#### **auto6: yes**

启用 IPv6 自动配置。默认情况下，NetworkManager 使用路由器公告。如果路由器宣布 **managed** 标记，则 NetworkManager 会从 DHCPv6 服务器请求 IPv6 地址和前缀。

#### **mode: <bond\_mode>**

设置绑定模式。可能的值有：

- **balance-rr**（默认）
- **active-backup**
- **balance-xor**
- **broadcast**
- **802.3ad**
- **balance-tlb**
- **balance-alb**

根据您设置的模式，您需要在 playbook 中设置额外的变量。

有关 playbook 中使用的所有变量的详情，请查看控制节点上的 `/usr/share/ansible/roles/rhel-system-roles.network/README.md` 文件。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不能防止错误的、但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

## 验证

- 从其中一个网络设备临时拔掉网线，并检查绑定中的其他设备是否在处理流量。

请注意，无法使用软件工具正确测试链路失败事件。停用连接的工具（如 `nmcli`），只显示绑定驱动程序可以处理端口配置的更改，而不是实际的链接失败事件。

## 其他资源

- `/usr/share/ansible/roles/rhel-system-roles.network/README.md` 文件
- `/usr/share/doc/rhel-system-roles/network/` 目录

## 3.9. 创建网络绑定以便在不中断 VPN 的情况下在以太网和无线连接间进行切换

将工作站连接到公司网络的 RHEL 用户通常会使用 VPN 访问远程资源。然而，如果工作站在以太网和 Wi-Fi 连接间切换，例如：如果您是从带以太网连接的 docking 站中释放的笔记本电脑，VPN 连接就中断。要避免这个问题，您可以在 **active-backup** 模式中创建使用以太网和 Wi-Fi 连接的网络绑定。

### 先决条件

- 主机包含以太网和 Wi-Fi 设备。
- 已创建以太网和 Wi-Fi 网络管理器连接配置集，且两个连接都可以独立工作。  
此流程使用以下连接配置文件来创建名为 **bond0** 的网络绑定：
  - 与 `enp1s0u1` 以太网设备关联的 **Docking\_station**
  - **Wi-Fi** 与 `wlp1s0` Wi-Fi 设备关联

### 流程

1. 在 **active-backup** 模式中创建一个绑定接口：

```
nmcli connection add type bond con-name bond0 ifname bond0 bond.options "mode=active-backup"
```

这个命令将接口和连接配置文件命名为 **bond0**。

## 2. 配置绑定的 IPv4 设置：

- 如果您的网络中的 DHCP 服务器为主机分配 IPv4 地址，则不需要任何操作。
- 如果您的本地网络需要静态 IPv4 地址，请将地址、网络掩码、默认网关、DNS 服务器和 DNS 搜索域设为 **bond0** 连接：

```
nmcli connection modify bond0 ipv4.addresses '192.0.2.1/24'
nmcli connection modify bond0 ipv4.gateway '192.0.2.254'
nmcli connection modify bond0 ipv4.dns '192.0.2.253'
nmcli connection modify bond0 ipv4.dns-search 'example.com'
nmcli connection modify bond0 ipv4.method manual
```

## 3. 配置绑定的 IPv6 设置：

- 如果您的网络中的路由器或者 DHCP 服务器为主机分配 IPv6 地址，则不需要任何操作。
- 如果您的本地网络需要静态 IPv6 地址，请将地址、网络掩码、默认网关、DNS 服务器和 DNS 搜索域设为 **bond0** 连接：

```
nmcli connection modify bond0 ipv6.addresses '2001:db8:1::1/64'
nmcli connection modify bond0 ipv6.gateway '2001:db8:1::fffe'
nmcli connection modify bond0 ipv6.dns '2001:db8:1::fffd'
nmcli connection modify bond0 ipv6.dns-search 'example.com'
nmcli connection modify bond0 ipv6.method manual
```

## 4. 显示连接配置文件：

```
nmcli connection show
NAME UUID TYPE DEVICE
Docking_station 256dd073-fecc-339d-91ae-9834a00407f9 ethernet enp11s0u1
Wi-Fi 1f1531c7-8737-4c60-91af-2d21164417e8 wifi wlp1s0
...
```

下一步需要连接配置集的名称和以太网设备名称。

## 5. 为绑定分配以太网连接的配置：

```
nmcli connection modify Docking_station master bond0
```

## 6. 为绑定分配 Wi-Fi 连接的连接配置集：

```
nmcli connection modify Wi-Fi master bond0
```

## 7. 如果您的 Wi-Fi 网络使用 MAC 过滤来只允许列表中的 MAC 地址访问网络，请配置 NetworkManager 将活跃端口的 MAC 地址动态分配给绑定：

```
nmcli connection modify bond0 +bond.options fail_over_mac=1
```

使用这个设置时，您必须将 Wi-Fi 设备的 MAC 地址设置为 allow 列表，而不是以太网和 Wi-Fi 设备的 MAC 地址。

- 将与以太连接关联的设备设置为绑定的主设备：

```
nmcli con modify bond0 +bond.options "primary=enp11s0u1"
```

使用这个设置时，如果可用，绑定总是使用以太网连接。

- 配置当 **bond0** 设备激活时，NetworkManager 会自动激活端口：

```
nmcli connection modify bond0 connection.autoconnect-slaves 1
```

- 激活 **bond0** 连接：

```
nmcli connection up bond0
```

## 验证

- 显示当前激活的设备，绑定及其端口的状态：

```
cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: fault-tolerance (active-backup) (fail_over_mac active)
Primary Slave: enp11s0u1 (primary_reselect always)
Currently Active Slave: enp11s0u1
MII Status: up
MII Polling Interval (ms): 1
Up Delay (ms): 0
Down Delay (ms): 0
Peer Notification Delay (ms): 0

Slave Interface: enp11s0u1
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 00:53:00:59:da:b7
Slave queue ID: 0

Slave Interface: wlp1s0
MII Status: up
Speed: Unknown
Duplex: Unknown
Link Failure Count: 2
Permanent HW addr: 00:53:00:b3:22:ba
Slave queue ID: 0
```

## 其它资源

- [配置以太网连接](#)
- [管理 Wi-Fi 连接](#)
- [配置网络绑定](#)

## 3.10. 不同的网络绑定模式

Linux 绑定驱动程序提供链路聚合。绑定是并行封装多个网络接口的过程，以提供单个逻辑绑定接口。绑定接口的操作取决于也称为模式的绑定策略。不同的模式提供负载均衡或热待机服务。

Linux 绑定驱动程序支持以下模式：

### Balance-rr (模式 0)

**balance-rr** 使用循环算法，它按顺序将数据包从第一个可用端口传输到最后一个端口。这个模式提供负载平衡和容错。

这个模式需要切换端口聚合组（也称为 EtherChannel 或类似的端口分组）。EtherChannel 是一个端口链路聚合技术，用于将多个物理以太网链接分组到一个逻辑以太网链接中。

这个模式的缺陷在于它不适用于大量工作负载，以及 TCP 吞吐量或排序数据包交付非常重要。

### Active-backup (模式 1)

**Active-backup** 使用策略来确定在绑定中只有一个端口活跃。这个模式提供容错功能，不需要任何交换机配置。

如果活动端口失败，则备用端口将变为活动状态。绑定会向网络发送大量地址解析协议 (ARP) 响应。gratuitous ARP 强制执行 ARP 帧的接收器，以更新它们的转发表。**Active-backup** 模式传输一个 gratuitous ARP，宣布为主机维护连接的新路径。

**primary** 选项定义绑定接口的首选端口。

### Balance-xor (模式 2)

**balance-xor** 使用所选传输哈希策略来发送数据包。这个模式提供负载平衡、容错和需要切换配置来设置 Etherchannel 或类似的端口分组。

要改变数据包传输和平衡传输，此模式使用 **xmit\_hash\_policy** 选项。根据接口上流量的源或目的地，接口需要额外的负载均衡配置。请参阅 [xmit\\_hash\\_policy bonding 参数](#)。

### Broadcast (模式 3)

**Broadcast** 使用在所有接口上传输每个数据包的策略。这个模式提供容错，需要交换机配置来设置 EtherChannel 或类似的端口分组。

这个模式的缺陷在于它不适用于大量工作负载，以及 TCP 吞吐量或排序数据包交付非常重要。

### 802.3ad (模式 4)

**802.3ad** 使用同名的 IEEE 标准动态链路聚合策略。此模式提供容错功能。这个模式需要切换配置来设置链路聚合控制协议 (LACP) 端口分组。

这个模式会创建聚合组，它们共享相同的速度和双工设置，并使用活跃聚合器中的所有端口。根据接口上流量的源或目的地，此模式需要额外的负载平衡配置。

默认情况下，传出流量的端口选择取决于传输哈希策略。使用传输哈希策略的 **xmit\_hash\_policy** 选项更改端口选择和平衡传输。

**802.3ad** 和 **Balance-xor** 之间的差别是合规性。**802.3ad** 策略在端口聚合组之间协商 LACP。请参阅 [xmit\\_hash\\_policy bonding 参数](#)

### Balance-tlb (模式 5)

**balance-tlb** 使用传输负载均衡策略。这个模式提供容错、负载均衡和建立不需要任何交换机支持的频道绑定。

活动端口接收传入流量。如果活动端口失败，另一个则是接管故障端口的 MAC 地址。要确定哪个接口处理传出流量，请使用以下模式之一：

- 值 **0**：使用哈希分发策略在不进行负载均衡的情况下分发流量
- 值 **1**：利用负载均衡将流量分配到每个端口  
使用 bonding 选项 **tlb\_dynamic\_lb=0**，此绑定模式使用 **xmit\_hash\_policy** bonding 选项来均衡传输。**primary** 选项定义绑定接口的首选端口。

请参阅 [xmit\\_hash\\_policy bonding 参数](#)。

### Balance-alb (模式 6)

**balance-alb** 使用自适应负载平衡策略。这个模式提供容错、负载平衡，且不需要任何特殊交换机支持。

这个模式包括平衡传输负载均衡(**balance-tlb**)和 IPv4 和 IPv6 流量的接收负载均衡。绑定会截获本地系统发送的 ARP 回复，并覆盖绑定中某个端口的源硬件地址。ARP 协商管理接收负载平衡。因此，不同的端口为服务器使用不同的硬件地址。

**primary** 选项定义绑定接口的首选端口。使用 bonding 选项 **tlb\_dynamic\_lb=0**，此绑定模式使用 **xmit\_hash\_policy** bonding 选项来均衡传输。请参阅 [xmit\\_hash\\_policy bonding 参数](#)。

## 其他资源

- [/usr/share/doc/kernel-doc-<version>/Documentation/networking/bonding.rst](#) 由 **kernel-doc** 软件包提供
- [/usr/share/doc/kernel-doc-<version>/Documentation/networking/bonding.txt](#) 由 **kernel-doc** 软件包提供
- [与虚拟机客户机或容器连接的网桥一起使用的绑定模式](#) (红帽知识库)
- [How are the values for different policies in "xmit\\_hash\\_policy" bonding parameter calculated?](#) (红帽知识库)

## 3.11. XMIT\_HASH\_POLICY BONDING 参数

**xmit\_hash\_policy** 负载均衡参数在 **balance-xor**、**802.3ad**、**balance-alb** 和 **balance-tlb** 模式中选择节点选择的传输散列策略。如果 **tlb\_dynamic\_lb** 参数为 **0**，则只适用于模式 5 和 6。此参数可能的值是 **layer2**, **layer2+3**, **layer3+4**, **encap2+3**, **encap3+4**, 和 **vlan+srcmac**。

详情请查看表：

| 策略或网络层 | Layer2 | Layer2+3 | Layer3+4 | encap2+3 | encap3+4 | VLAN+src mac |
|--------|--------|----------|----------|----------|----------|--------------|
|        |        |          |          |          |          |              |

|                  |                              |                           |                                        |                                                                                                   |                                                                                    |                                                                                 |
|------------------|------------------------------|---------------------------|----------------------------------------|---------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------|---------------------------------------------------------------------------------|
| <b>使用</b>        | 源和目的地 MAC 地址和以太网协议类型的 XOR    | 源和目标 MAC 地址和 IP 地址的 XOR   | 源和目标端口和 IP 地址的 XOR                     | 支持的隧道内的目的地 MAC 地址和 IP 地址的 XOR, 如虚拟可扩展局域网 (VXLAN)。此模式依赖于 <code>skb_flow_dissect()</code> 函数来获取标头字段 | 受支持的隧道内的目标端口和 IP 地址的 XOR, 如 VXLAN。此模式依赖于 <code>skb_flow_dissect()</code> 函数来获取标头字段 | VLAN ID 和源 MAC 厂商和源 MAC 设备的 XOR                                                 |
| <b>流量放置</b>      | 在同一个底层网络接口上到特定网络对等的所有流量      | 同一底层网络接口上特定 IP 地址的所有流量    | 同一底层网络接口上特定 IP 地址和端口的所有流量              |                                                                                                   |                                                                                    |                                                                                 |
| <b>主要选择</b>      | 如果网络流量在同一广播域中的这个系统和多个其他系统之间  | 如果此系统和多个其他系统间的网络流量会通过默认网关 | 如果此系统和其他系统之间的网络流量使用相同的 IP 地址, 但会经历多个端口 | 封装的流量在源系统和多个其它系统中使用多个 IP 地址                                                                       | 封装的流量是源系统和其它使用多个端口号的系统间                                                            | 如果绑定承载来自多个容器或虚拟机 (VM) 的网络流量, 它会将其 MAC 地址直接公开给外部网络, 如桥接网络, 您无法配置模式 2 或模式 4 的交换机。 |
| <b>辅助选择</b>      | 如果网络流量主要是此系统和默认网关后面的多个其他系统之间 | 如果网络流量主要是此系统和另一个系统间的      |                                        |                                                                                                   |                                                                                    |                                                                                 |
| <b>Compliant</b> | 802.3ad                      | 802.3ad                   | Not 802.3ad                            |                                                                                                   |                                                                                    |                                                                                 |
| <b>默认策略</b>      | 如果没有提供配置, 则这是默认策略            | 对于非 IP 流量, 公式与 L2 传输策略相同  | 对于非 IP 流量, 公式与 L2 传输策略相同               |                                                                                                   |                                                                                    |                                                                                 |

## 第 4 章 配置 NIC TEAM

网络接口控制器(NIC)team 是一种组合或聚合物理和虚拟网络接口的方法，以提供具有高吞吐量或冗余的逻辑接口。NIC team 使用小内核模块来实现数据包流的快速处理和用于其他任务的用户空间服务。这样，NIC team 是一种用于负载平衡和冗余要求的易于扩展和伸缩的解决方案。

Red Hat Enterprise Linux 为管理员提供不同的选项来配置团队设备。例如：

- 使用 **nmcli** 使用命令行配置团队连接。
- 使用 RHEL web 控制台使用 Web 浏览器配置组连接。
- 使用 **nm-connection-editor** 应用程序在图形界面中配置组连接。



### 重要

NIC team 在 Red Hat Enterprise Linux 9 中已弃用。如果您计划将服务器升级到将来的 RHEL 版本，请考虑使用内核绑定驱动程序作为替代方案。详情请参阅 [配置网络绑定](#)。

### 4.1. 了解控制器和端口接口的默认行为

在使用 **NetworkManager** 服务管理或排除团队或绑定端口接口故障时，请考虑以下默认行为：

- 启动控制器接口不会自动启动端口接口。
- 启动端口接口总会启动控制器接口。
- 停止控制器接口也会停止端口接口。
- 没有端口的控制器可以启动静态 IP 连接。
- 没有端口的控制器在启动 DHCP 连接时会等待端口。
- 当您添加具有载体的端口时，等待端口且具有 DHCP 连接的控制器会完成。
- 当您添加没有载体的端口时，具有等待端口的 DHCP 连接的控制器继续等待。

### 4.2. 了解 TEAMD 服务、运行程序和 LINK-WATCHERS

团队服务 **teamd** 控制团队驱动程序的一个实例。这个驱动的实例添加硬件设备驱动程序实例组成一个网络接口组。团队驱动程序向内核提供一个网络接口，如 **team0**。

**teamd** 服务对所有团队方法实现通用逻辑。这些功能对不同的负载共享和备份方法（如循环）是唯一的，并由称为 **runners** 的单独的代码单元来实现。管理员以 JavaScript 对象表示法(JSON)格式指定runners，在创建实例时，JSON 代码被编译到 **teamd** 实例中。另外，在使用 **NetworkManager** 时，您可以在 **team.runner** 参数中设置 runner，**NetworkManager** 会自动创建对应的 JSON 代码。

可用的 runner 如下：

- **broadcast**：转换所有端口上的数据。
- **roundrobin**：依次转换所有端口上的数据。
- **activebackup**：转换一个端口上的数据，而其他端口上的数据则作为备份保留。

- **loadbalance**：转换所有具有活跃的 Tx 负载均衡和基于 Berkeley 数据包过滤器(BPF)的 Tx 端口选择器的端口上的数据。
- **random**：转换随机选择的端口上的数据。
- **lacp**：实现 802.3ad 链路聚合控制协议(LACP)。

**teamd** 服务使用链路监视器来监控从属设备的状态。可用的 link-watchers 如下：

- **ethtool**：libteam 库使用 **ethtool** 工具来监视链接状态的变化。这是默认的 link-watcher。
- **arp\_ping**：libteam 库使用 **arp\_ping** 工具来监控使用地址解析协议(ARP)的远端硬件地址是否存在。
- **nsna\_ping**：在 IPv6 连接上，libteam 库使用来自 IPv6 邻居发现协议的邻居广告和邻居请求功能来监控邻居接口的存在。

每个 runner 都可以使用任何链接监视器，但 **lacp** 除外。此 runner 只能使用 **ethtool** 链接监视器。

## 4.3. 使用 NMCLI 配置 NIC TEAM

要在命令行上配置网络接口控制器(NIC)team，请使用 **nmcli** 工具。



### 重要

NIC team 在 Red Hat Enterprise Linux 9 中已弃用。如果您计划将服务器升级到将来的 RHEL 版本，请考虑使用内核绑定驱动程序作为替代方案。详情请参阅 [配置网络绑定](#)。

### 先决条件

- 已安装 **teamd** 和 **NetworkManager-team** 软件包。
- 在服务器中安装两个或者两个以上物理或者虚拟网络设备。
- 要将以太网设备用作组的端口，必须在服务器中安装物理或者虚拟以太网设备并连接到交换机。
- 要使用 **bond**、**bridge** 或 **VLAN** 设备作为团队的端口，您可以在创建团队时创建这些设备，或者预先创建它们，如下所述：
  - [使用 nmcli 配置网络绑定](#)
  - [使用 nmcli 配置网桥](#)
  - [使用 nmcli 配置 VLAN 标记](#)

### 流程

#### 1. 创建团队接口：

```
nmcli connection add type team con-name team0 ifname team0 team.runner activebackup
```

此命令创建一个名为 **team0** 的 NIC team，它使用 **activebackup** runner。

#### 2. 可选：设置链接监视程序。例如，要在 **team0** 连接配置文件中设置 **ethtool** 链接监视器：

```
nmcli connection modify team0 team.link-watchers "name=ethtool"
```

链路监视器支持不同的参数。要为链路监视器设置参数，请在 **name** 属性中以空格分隔的方式来指定它们。请注意，**name** 属性必须用引号括起。例如，要使用 **ethtool** 链接监视器，并将其 **delay-up** 参数设置为 2500 毫秒（2.5 秒）：

```
nmcli connection modify team0 team.link-watchers "name=ethtool delay-up=2500"
```

要设置多个链路监视器，每个都使用特定的参数，不同的连接监视器以逗号分隔。以下示例使用 **delay-up** 参数设置 **ethtool** 链接监视器，使用 **source-host** 和 **target-host** 参数设置 **arp\_ping** 链路监视器：

```
nmcli connection modify team0 team.link-watchers "name=ethtool delay-up=2,
name=arp_ping source-host=192.0.2.1 target-host=192.0.2.2"
```

### 3. 显示网络接口，并记录您要添加到团队中的接口名称：

```
nmcli device status
DEVICE TYPE STATE CONNECTION
enp7s0 ethernet disconnected --
enp8s0 ethernet disconnected --
bond0 bond connected bond0
bond1 bond connected bond1
...
...
```

在本例中：

- 没有配置 **enp7s0** 和 **enp8s0**。要将这些设备用作端口，请在下一步中添加连接配置集。请注意，您只能在没有分配给任何连接的团队中使用以太网接口。
- bond0** 和 **bond1** 已有连接配置文件。要将这些设备用作端口，请在下一步中修改其配置集。

### 4. 为团队分配端口接口：

- 如果没有配置您要分配给团队的接口，为其创建新的连接配置集：

```
nmcli connection add type ethernet slave-type team con-name team0-port1
ifname enp7s0 master team0
nmcli connection add type ethernet slave--type team con-name team0-port2
ifname enp8s0 master team0
```

这些命令为 **enp7s0** 和 **enp8s0** 创建配置文件，并将它们添加到 **team0** 连接中。

- 为团队分配现有连接配置集：

- 将这些连接的 **master** 参数设置为 **team0**：

```
nmcli connection modify bond0 master team0
nmcli connection modify bond1 master team0
```

这些命令将名为 **bond0** 和 **bond1** 的现有连接配置文件分配给 **team0** 连接。

- 重新激活连接：

```
nmcli connection up bond0
nmcli connection up bond1
```

## 5. 配置 IPv4 设置：

- 要为 **team0** 连接设置静态 IPv4 地址、网络掩码、默认网关和 DNS 服务器设置，请输入：

```
nmcli connection modify team0 ipv4.addresses '192.0.2.1/24' ipv4.gateway
'192.0.2.254' ipv4.dns '192.0.2.253' ipv4.dns-search 'example.com' ipv4.method
manual
```

- 要使用 DHCP，不需要任何操作。
- 如果您计划将此团队设备用作其它设备的端口，则不需要任何操作。

## 6. 配置 IPv6 设置：

- 要为 **team0** 连接设置静态 IPv6 地址、网络掩码、默认网关和 DNS 服务器设置，请输入：

```
nmcli connection modify team0 ipv6.addresses '2001:db8:1::1/64' ipv6.gateway
'2001:db8:1::ffffe' ipv6.dns '2001:db8:1::ffffd' ipv6.dns-search 'example.com'
ipv6.method manual
```

- 如果您计划将此团队设备用作其它设备的端口，则不需要任何操作。
- 要使用无状态地址自动配置(SLAAC)，不需要采取任何操作。

## 7. 激活连接：

```
nmcli connection up team0
```

## 验证

- 显示团队状态：

```
teamdctl team0 state
setup:
runner: activebackup
ports:
enp7s0
link watches:
link summary: up
instance[link_watch_0]:
name: ethtool
link: up
down count: 0
enp8s0
link watches:
link summary: up
instance[link_watch_0]:
name: ethtool
link: up
down count: 0
runner:
active port: enp7s0
```

■ 在这个示例中，两个端口都是上线的。

## 其他资源

- 配置 NetworkManager 以避免使用特定配置集提供默认网关
- 了解 teamd 服务、运行程序和 link-watchers
- 您系统上的 **nm-settings (5)** 和 **teamd.conf (5)** 手册页

## 4.4. 使用 RHEL WEB 控制台配置 NIC TEAM

如果您希望使用基于 Web 浏览器的界面管理网络设置，请使用 RHEL web 控制台配置网络接口控制器(NIC)team。



### 重要

NIC team 已在 Red Hat Enterprise Linux 9 中弃用。如果您计划将服务器升级到将来的 RHEL 版本，请考虑使用内核绑定驱动程序作为替代方案。详情请参阅 [配置网络绑定](#)。

### 先决条件

- 已安装 **teamd** 和 **NetworkManager-team** 软件包。
- 在服务器中安装两个或者两个以上物理或者虚拟网络设备。
- 要将以太网设备用作组的端口，必须在服务器中安装物理或者虚拟以太网设备并连接到交换机。
- 要将 bond、bridge 或 VLAN 设备用作团队的端口，请预先创建它们，如下所述：
  - 使用 RHEL web 控制台配置网络绑定
  - 使用 RHEL web 控制台配置网桥
  - 使用 RHEL web 控制台配置 VLAN 标记
- 您已安装了 RHEL 8 web 控制台。
- 您已启用了 cockpit 服务。
- 您的用户帐户被允许登录到 web 控制台。  
具体步骤请参阅 [安装并启用 Web 控制台](#)。

### 步骤

1. 登录到 RHEL 8 web 控制台。  
详情请参阅 [登录到 web 控制台](#)。
2. 在屏幕左侧的导航中选择 **Networking** 选项卡。
3. 在 **Interfaces** 部分点 **Add team**。
4. 输入您要创建的团队设备名称。

5. 选择应该是团队端口的接口。
6. 选择团队的运行程序。  
如果您选择 **Load balancing** 或 **802.3ad LACP**, Web 控制台会显示额外的 **Balancer** 字段。
7. 设置链接监视器：
  - 如果您选择 **Ethtool**, 请设置链接并关闭延迟。
  - 如果您设置了 **ARP ping** 或 **NSNA ping**, 还要设置 ping 间隔并 ping 目标。

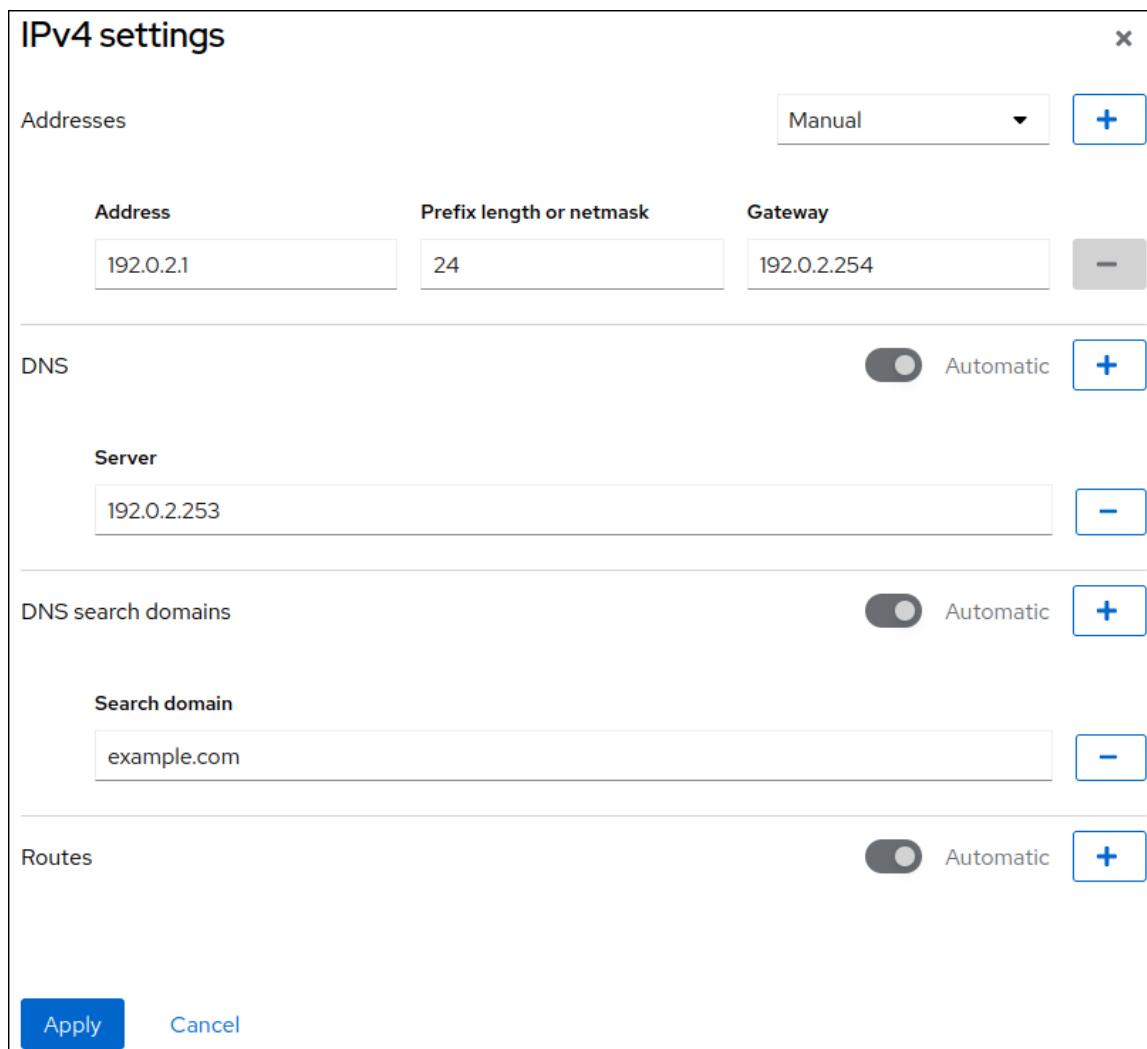
**Team settings**

|                                                                            |                                                                                          |
|----------------------------------------------------------------------------|------------------------------------------------------------------------------------------|
| Name                                                                       | team0                                                                                    |
| Ports                                                                      | <input checked="" type="checkbox"/> enp7s0<br><input checked="" type="checkbox"/> enp8s0 |
| Runner                                                                     | Active backup                                                                            |
| Link watch                                                                 | Ethtool                                                                                  |
| Link up delay                                                              | 0                                                                                        |
| Link down delay                                                            | 0                                                                                        |
| <input type="button" value="Apply"/> <input type="button" value="Cancel"/> |                                                                                          |

8. 点应用。
9. 默认情况下, 团队使用动态 IP 地址。如果要设置静态 IP 地址 :
  - a. 在 **Interfaces** 部分点团队的名称。
  - b. 点您要配置的协议旁的 **Edit**。
  - c. 选择 **Addresses** 旁的 **Manual**, 并输入 IP 地址、前缀和默认网关。
  - d. 在 **DNS** 部分, 点 + 按钮, 并输入 DNS 服务器的 IP 地址。重复此步骤来设置多个 DNS 服务 器。

e. 在 **DNS search domains** 部分中，点 + 按钮并输入搜索域。

f. 如果接口需要静态路由，请在 **Routes** 部分配置它们。



g. 点 应用

## 验证

1. 在屏幕左侧的导航中选择 **Networking** 选项卡，并检查接口上是否有传入和传出流量。

| Interfaces |              | <input type="button" value="Add bond"/> | <input type="button" value="Add team"/> | <input type="button" value="Add bridge"/> | <input type="button" value="Add VLAN"/> |
|------------|--------------|-----------------------------------------|-----------------------------------------|-------------------------------------------|-----------------------------------------|
| Name       | IP address   | Sending                                 | Receiving                               |                                           |                                         |
| team0      | 192.0.2.1/24 | 1.11 Mbps                               | 61.2 Mbps                               |                                           |                                         |

2. 显示团队状态：

```
teamdctl team0 state
setup:
runner: activebackup
ports:
 enp7s0
link watches:
 link summary: up
```

```

instance[link_watch_0]:
 name: ethtool
 link: up
 down count: 0
enp8s0
link watches:
 link summary: up
instance[link_watch_0]:
 name: ethtool
 link: up
 down count: 0
runner:
 active port: enp7s0

```

在这个示例中，两个端口都是上线的。

## 其他资源

- [了解 teamd 服务、运行程序和 link-watchers](#)

## 4.5. 使用 NM-CONNECTION-EDITOR 配置 NIC TEAM

如果使用带有图形界面的 Red Hat Enterprise Linux，您可以使用 **nm-connection-editor** 应用程序配置网络接口控制器(NIC)team。

请注意：**nm-connection-editor** 只能向团队添加新端口。要使用现有连接配置文件作为端口，请使用 **nmcli** 工具创建 team，如 [使用 nmcli 配置 NIC team](#) 中所述。



### 重要

NIC team 已在 Red Hat Enterprise Linux 9 中弃用。如果您计划将服务器升级到将来的 RHEL 版本，请考虑使用内核绑定驱动程序作为替代方案。详情请参阅 [配置网络绑定](#)。

## 先决条件

- 已安装 **teamd** 和 **NetworkManager-team** 软件包。
- 在服务器中安装两个或者两个以上物理或者虚拟网络设备。
- 要将以太网设备用作组的端口，必须在服务器中安装物理或者虚拟以太网设备。
- 要使用 team、bond 或 VLAN 设备作为团队的端口，请确保这些设备还没有配置。

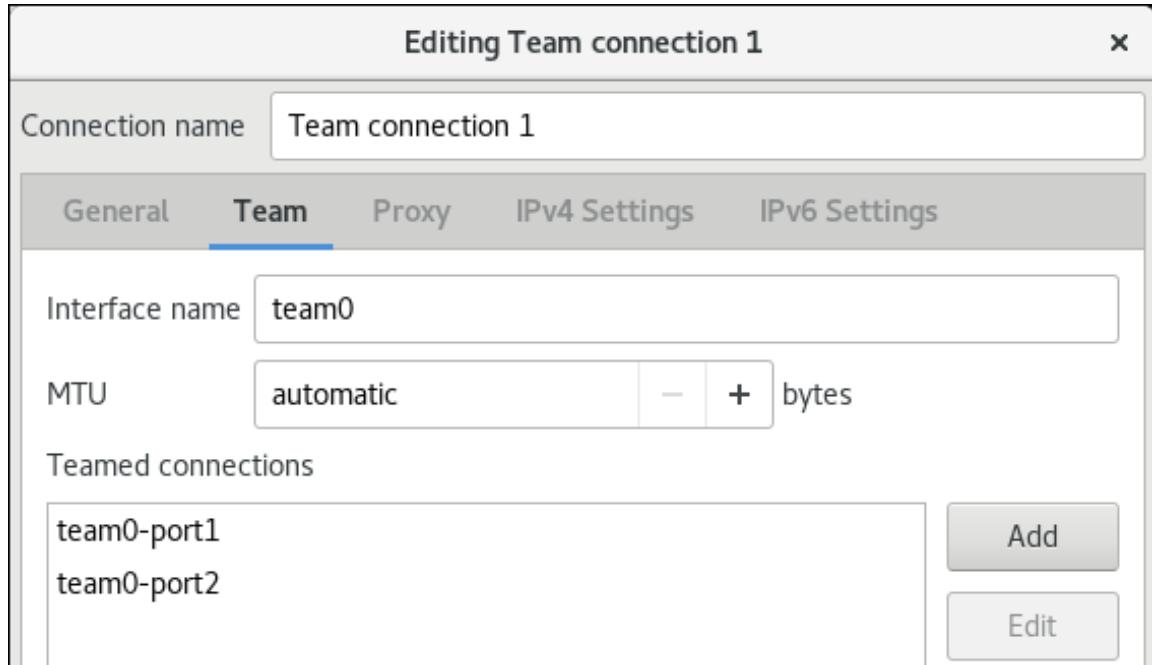
## 步骤

1. 打开一个终端，输入 **nm-connection-editor**：

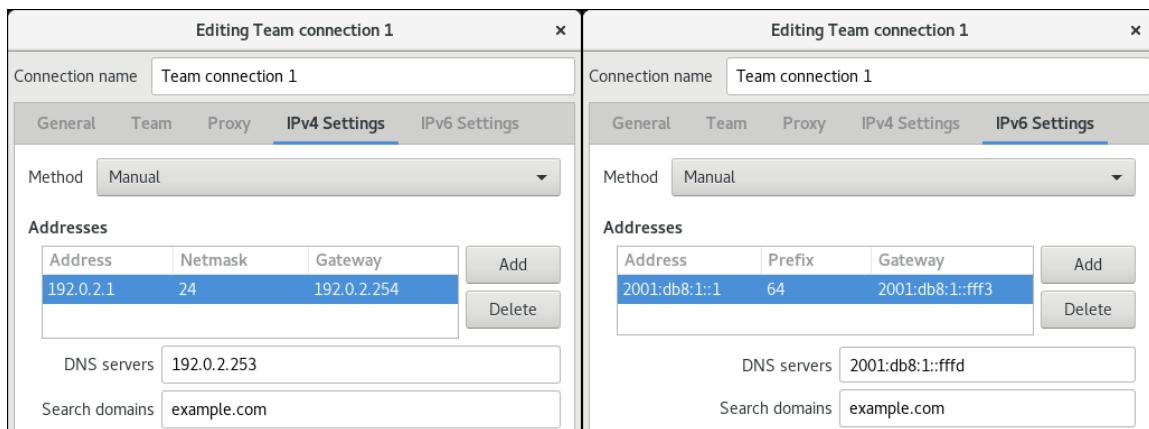
```
$ nm-connection-editor
```

2. 点击 + 按钮来添加一个新的连接。
3. 选择 **Team** 连接类型，然后单击 **Create**。
4. 在 **Team** 选项卡中：

- a. 可选：在 **Interface name** 字段中设置团队接口的名称。
- b. 点 **Add** 按钮为网络接口添加新连接配置集，并将配置集作为端口添加到团队。
  - i. 选择接口的连接类型。例如，为有线连接选择 **Ethernet**。
  - ii. 可选：为端口设置连接名称。
  - iii. 如果您为以太网设备创建连接配置文件，请打开 **Ethernet** 选项卡，在 **Device** 字段中选择您要作为端口添加到团队的网络接口。如果您选择了不同的设备类型，请相应地进行配置。请注意，您只能在没有分配给任何连接的团队中使用以太网接口。
  - iv. 点 **Save**。
- c. 对您要添加到团队的每个接口重复前面的步骤。



- d. 点 **Advanced** 按钮将高级选项设置为团队连接。
  - i. 在 **Runner** 选项卡中，选择 runner。
  - ii. 在 **Link Watcher** 选项卡中，设置链接监视器及其可选设置。
  - iii. 点确定。
5. 在 **IPv4 Settings** 和 **IPv6 Settings** 标签页中配置 IP 地址设置：
  - 如果您计划将此网桥设备用作其它设备的端口，请将 **Method** 字段设置为 **Disabled**。
  - 要使用 DHCP，请将 **Method** 字段保留为默认值 **Automatic (DHCP)**。
  - 要使用静态 IP 设置，请将 **Method** 字段设置为 **Manual**，并相应地填写字段：



6. 点 Save。

7. 关闭 nm-connection-editor。

## 验证

- 显示团队状态：

```
teamdctl team0 state
setup:
 runner: activebackup
ports:
 enp7s0
 link watches:
 link summary: up
 instance[link_watch_0]:
 name: ethtool
 link: up
 down count: 0
 enp8s0
 link watches:
 link summary: up
 instance[link_watch_0]:
 name: ethtool
 link: up
 down count: 0
runner:
 active port: enp7s0
```

## 其它资源

- 使用 nm-connection-editor 配置网络绑定
- 使用 nm-connection-editor 配置 NIC team
- 使用 nm-connection-editor 配置 VLAN 标记
- 配置 NetworkManager 以避免使用特定配置集提供默认网关
- 了解 teamd 服务、运行程序和 link-watchers
- 重启 NetworkManager 服务后 NetworkManager 会复制连接 (红帽知识库)

## 第 5 章 配置 VLAN 标记

Virtual Local Area Network (VLAN) 是物理网络中的一个逻辑网络。当 VLAN 接口通过接口时，VLAN 接口标签带有 VLAN ID 的数据包，并删除返回的数据包的标签。您可以在另一个接口（如以太网、绑定、团队或桥接设备）上创建 VLAN 接口。这些接口称为 **父接口**。

Red Hat Enterprise Linux 为管理员提供不同的选项来配置 VLAN 设备。例如：

- 使用 **nmcli** 使用命令行配置 VLAN 标记。
- 通过 RHEL web 控制台使用 Web 浏览器配置 VLAN 标记。
- 使用 **nmtui** 在基于文本的用户界面中配置 VLAN 标记。
- 使用 **nm-connection-editor** 应用程序在图形界面中配置连接。
- 使用 **nmstatectl** 通过 Nmstate API 配置连接。
- 使用 RHEL 系统角色在一个或多个主机上自动化 VLAN 配置。

### 5.1. 使用 NMCLI 配置 VLAN 标记

您可以使用 **nmcli** 实用程序在命令行中配置 Virtual Local Area Network (VLAN) 标记。

#### 先决条件

- 您计划用作虚拟 VLAN 接口的父接口支持 VLAN 标签。
- 如果您在绑定接口之上配置 VLAN：
  - 绑定的端口是上线的。
  - 这个绑定没有使用 **fail\_over\_mac=follow** 选项进行配置。VLAN 虚拟设备无法更改其 MAC 地址以匹配父设备的新 MAC 地址。在这种情况下，流量仍会与不正确的源 MAC 地址一同发送。
  - 这个绑定通常不会预期从 DHCP 服务器或 IPv6 自动配置获取 IP 地址。在创建绑定时通过设置 **ipv4.method=disable** 和 **ipv6.method=ignore** 选项来确保它。否则，如果 DHCP 或 IPv6 自动配置在一段时间后失败，接口可能会关闭。
- 主机连接到的交换机被配置为支持 VLAN 标签。详情请查看您的交换机文档。

#### 流程

1. 显示网络接口：

```
nmcli device status
DEVICE TYPE STATE CONNECTION
enp1s0 ethernet disconnected enp1s0
bridge0 bridge connected bridge0
bond0 bond connected bond0
...
```

2. 创建 VLAN 接口。例如，要创建一个使用 **enp1s0** 作为其父接口，使用 VLAN ID **10** 标记数据包，名为 **vlan10** 的 VLAN 接口，请输入：

```
nmcli connection add type vlan con-name vlan10 ifname vlan10 vlan.parent enp1s0
vlan.id 10
```

请注意，VLAN 必须在范围 **0** 到 **4094** 之间。

3. 默认情况下，VLAN 连接会继承上级接口的最大传输单元（MTU）。另外，还可设置不同的 MTU 值：

```
nmcli connection modify vlan10 ethernet.mtu 2000
```

4. 配置 IPv4 设置：

- 要为 **vlan10** 连接设置静态 IPv4 地址、网络掩码、默认网关和 DNS 服务器设置，请输入：

```
nmcli connection modify vlan10 ipv4.addresses '192.0.2.1/24' ipv4.gateway
'192.0.2.254' ipv4.dns '192.0.2.253' ipv4.method manual
```

- 要使用 DHCP，不需要任何操作。
- 如果您计划将此 VLAN 设备用作其它设备的端口，则不需要任何操作。

5. 配置 IPv6 设置：

- 要为 **vlan10** 连接设置静态 IPv6 地址、网络掩码、默认网关和 DNS 服务器设置，请输入：

```
nmcli connection modify vlan10 ipv6.addresses '2001:db8:1::1/32' ipv6.gateway
'2001:db8:1::ffff' ipv6.dns '2001:db8:1::ffff' ipv6.method manual
```

- 要使用无状态地址自动配置(SLAAC)，不需要采取任何操作。
- 如果您计划将此 VLAN 设备用作其它设备的端口，则不需要任何操作。

6. 激活连接：

```
nmcli connection up vlan10
```

## 验证

- 验证设置：

```
ip -d addr show wlan10
4: wlan10@enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
state UP group default qlen 1000
 link/ether 52:54:00:72:2f:6e brd ff:ff:ff:ff:ff:ff promiscuity 0
 vlan protocol 802.1Q id 10 <REORDER_HDR> numtxqueues 1 numrxqueues 1
 gso_max_size 65536 gso_max_segs 65535
 inet 192.0.2.1/24 brd 192.0.2.255 scope global noprefixroute wlan10
 valid_lft forever preferred_lft forever
 inet6 2001:db8:1::1/32 scope global noprefixroute
```

```

valid_lft forever preferred_lft forever
inet6 fe80::8dd7:9030:6f8e:89e6/64 scope link noprefixroute
 valid_lft forever preferred_lft forever

```

## 其他资源

- 您系统上的 **nm-settings (5)** 手册页

## 5.2. 使用 RHEL WEB 控制台配置 VLAN 标记

如果希望在 RHEL web 控制台中使用基于 Web 浏览器的界面管理网络设置，您可以配置 VLAN 标记。

### 先决条件

- 您计划用作虚拟 VLAN 接口的父接口支持 VLAN 标签。
- 如果您在绑定接口之上配置 VLAN：
  - 绑定的端口是上线的。
  - 这个绑定没有使用 **fail\_over\_mac=follow** 选项进行配置。VLAN 虚拟设备无法更改其 MAC 地址以匹配父设备的新 MAC 地址。在这种情况下，流量仍会与不正确的源 MAC 地址一同发送。
  - 这个绑定通常不会预期从 DHCP 服务器或 IPv6 自动配置获取 IP 地址。禁用 IPv4 和 IPv6 协议创建绑定以确保它。否则，如果 DHCP 或 IPv6 自动配置在一段时间后失败，接口可能会关闭。
- 主机连接到的交换机被配置为支持 VLAN 标签。详情请查看您的交换机文档。
- 您已安装了 RHEL 8 web 控制台。
- 您已启用了 cockpit 服务。
- 您的用户帐户被允许登录到 web 控制台。  
具体步骤请参阅[安装并启用 Web 控制台](#)。

### 流程

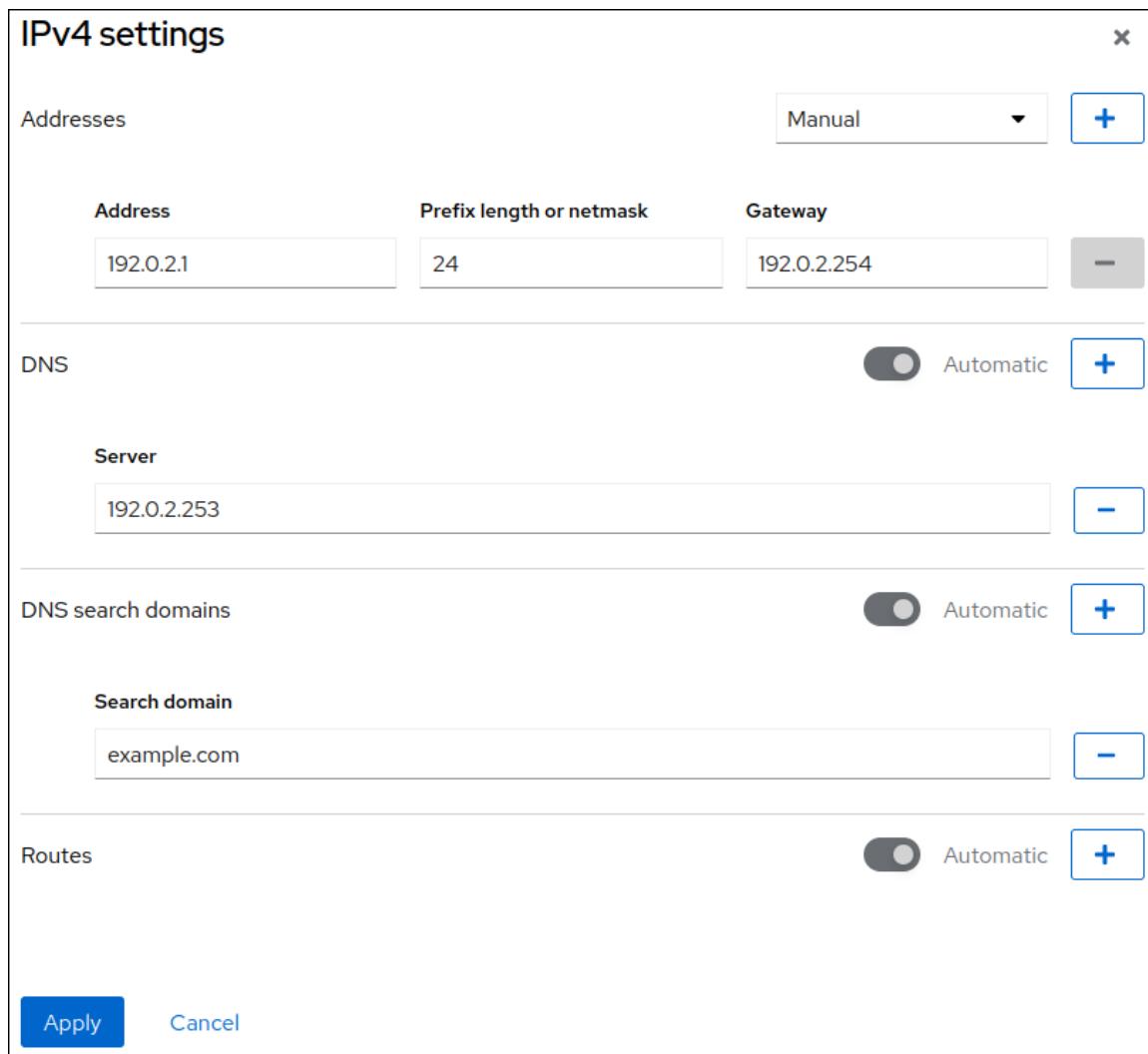
1. 登录到 RHEL 8 web 控制台。  
详情请参阅[登录到 web 控制台](#)。
2. 在屏幕左侧的导航中选择 **Networking** 选项卡。
3. 在 **Interfaces** 部分点 **Add VLAN**。
4. 选择父设备。
5. 输入 VLAN ID。
6. 输入 VLAN 设备的名称，或保留自动生成的名称。

VLAN settings

|         |           |
|---------|-----------|
| Parent  | enp1s0    |
| VLAN ID | 10        |
| Name    | enp1s0.10 |

**Apply** **Cancel**

7. 点应用。
8. 默认情况下，VLAN 设备使用动态 IP 地址。如果要设置静态 IP 地址：
  - a. 点 **Interfaces** 部分中的 VLAN 设备名称。
  - b. 点您要配置的协议旁的 **Edit**。
  - c. 选择 **Addresses** 旁的 **Manual**，并输入 IP 地址、前缀和默认网关。
  - d. 在 **DNS** 部分，点 **+** 按钮，并输入 DNS 服务器的 IP 地址。重复此步骤来设置多个 DNS 服务器。
  - e. 在 **DNS search domains** 部分中，点 **+** 按钮并输入搜索域。
  - f. 如果接口需要静态路由，请在 **Routes** 部分配置它们。



g. 点应用

## 验证

- 在屏幕左侧的导航中选择 **Networking** 选项卡，并检查接口上是否有传入和传出流量：

| Interfaces                |              | <a href="#">Add bond</a> | <a href="#">Add team</a> | <a href="#">Add bridge</a> | <a href="#">Add VLAN</a> |
|---------------------------|--------------|--------------------------|--------------------------|----------------------------|--------------------------|
| Name                      | IP address   | Sending                  | Receiving                |                            |                          |
| <a href="#">enp1s0.10</a> | 192.0.2.1/24 | 1.11 Mbps                | 61.2 Mbps                |                            |                          |

## 5.3. 使用 nmtui 配置 VLAN 标记

**nmtui** 应用程序为 NetworkManager 提供了一个基于文本的用户界面。您可以使用 **nmtui** 在没有图形界面的主机上配置 VLAN 标签。



## 注意

在 **nmtui** 中：

- 使用光标键导航。
- 选择一个按钮并按 **Enter** 键。
- 使用 **空格** 选择和清除复选框。
- 要返回上一个屏幕，请使用 **ESC**。

## 先决条件

- 您计划用作虚拟 VLAN 接口的父接口支持 VLAN 标签。
- 如果您在绑定接口之上配置 VLAN：
  - 绑定的端口是上线的。
  - 这个绑定没有使用 **fail\_over\_mac=follow** 选项进行配置。VLAN 虚拟设备无法更改其 MAC 地址以匹配父设备的新 MAC 地址。在这种情况下，流量仍会与不正确的源 MAC 地址一同发送。
  - 这个绑定通常不会预期从 DHCP 服务器或 IPv6 自动配置获取 IP 地址。在创建绑定时通过设置 **ipv4.method=disabled** 和 **ipv6.method=ignore** 选项来确保它。否则，如果 DHCP 或 IPv6 自动配置在一段时间后失败，接口可能会关闭。
- 主机连接到的交换机被配置为支持 VLAN 标签。详情请查看您的交换机文档。

## 步骤

1. 如果您不知道要在其上配置 VLAN 标签的网络设备名称，请显示可用的设备：

```
nmcli device status
DEVICE TYPE STATE CONNECTION
enp1s0 ethernet unavailable --
```

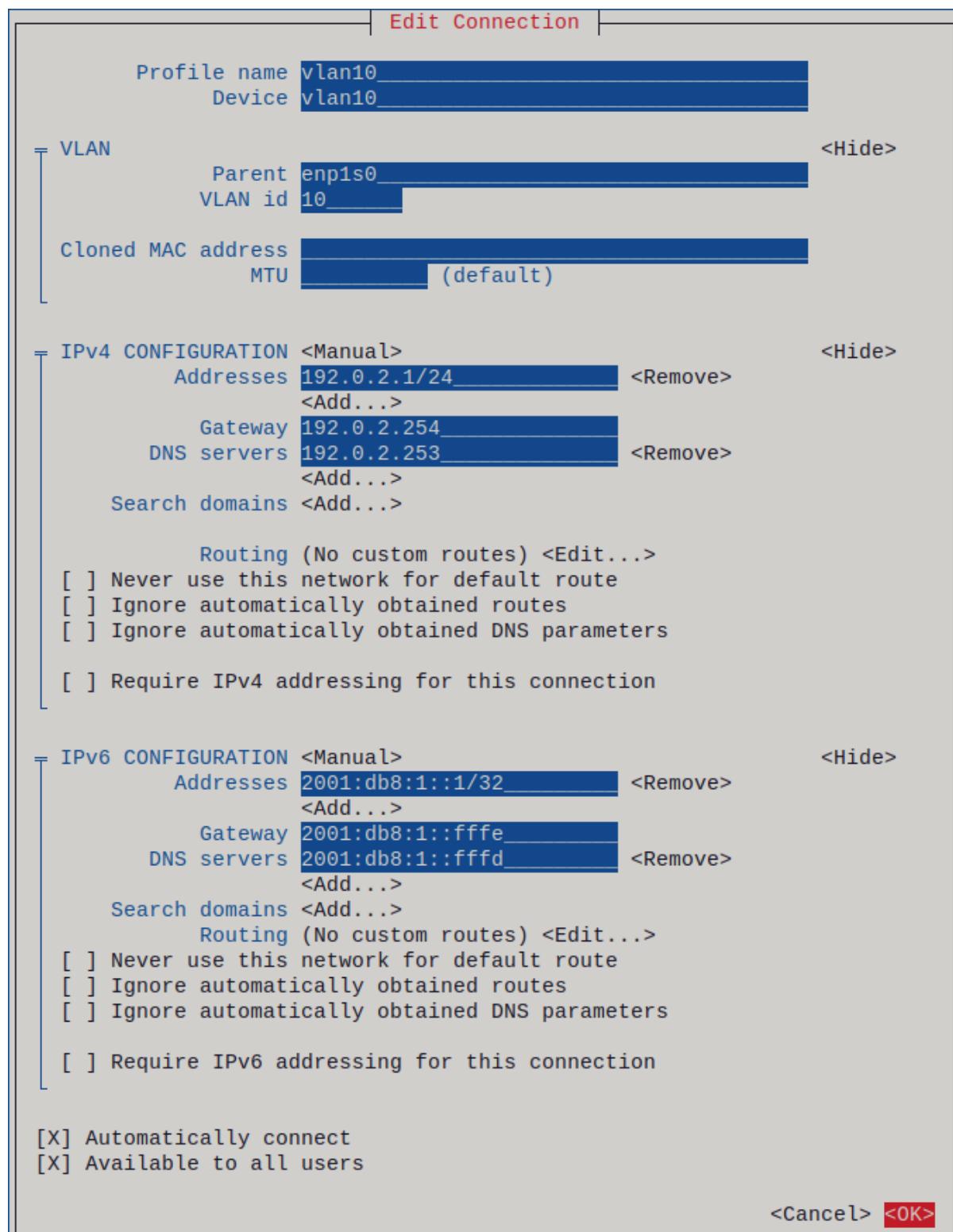
2. 启动 **nmtui**：

```
nmtui
```

3. 选择 **Edit a connection**，然后按 **Enter**。
4. 按 **Add**。
5. 从网络类型列表中选择 **VLAN**，然后按 **Enter** 键。
6. 可选：为要创建的 NetworkManager 配置文件输入一个名称。  
在有多个配置文件的主机上，有意义的名称可以更容易识别配置文件的用途。
7. 在 **Device** 字段中输入要创建的 VLAN 设备名称。
8. 在 **Parent** 字段中输入您要在其上配置 VLAN 标记的设备的名称。

9. 输入 VLAN ID。ID 必须在 **0** 到 **4094** 之间。
10. 根据您的环境，相应地在 **IPv4 configuration** 和 **IPv6 configuration** 配置区域中配置 IP 地址。为此，请按这些区域旁边的按钮，并选择：
  - **Disabled**，如果此 VLAN 设备不需要 IP 地址，或者您想要将其用作其它设备的端口。
  - **Automatic**，如果 DHCP 服务器或无状态地址自动配置(SLAAC)将 IP 地址动态分配给 VLAN 设备。
  - **Manual**，如果网络需要静态 IP 地址设置。在这种情况下，您必须填写更多字段：
    - i. 在您要配置的协议旁边按 **Show** 以显示其他字段。
    - ii. 按 **Addresses** 旁边的 **Add**，并输入无类别域间路由(CIDR)格式的 IP 地址和子网掩码。如果没有指定子网掩码，NetworkManager 会为 IPv4 地址设置 /**32** 子网掩码，为 IPv6 地址设置 /**64** 子网掩码。
    - iii. 输入默认网关的地址。
    - iv. 按 **DNS servers** 旁边的 **Add**，并输入 DNS 服务器地址。
    - v. 按 **Search domains** 旁边的 **Add**，并输入 DNS 搜索域。

图 5.1. 具有静态 IP 地址设置的 VLAN 连接示例



11. 按 **OK** 创建并自动激活新连接。
12. 按 **Back** 返回到主菜单。
13. 选择 **Quit**, 然后按 **Enter** 键关闭 **nmtui** 应用程序。

## 验证

- 验证设置：

```
ip -d addr show vlan10
4: vlan10@enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
state UP group default qlen 1000
 link/ether 52:54:00:72:2f:6e brd ff:ff:ff:ff:ff:ff promiscuity 0
 vlan protocol 802.1Q id 10 <REORDER_HDR> numtxqueues 1 numrxqueues 1
 gso_max_size 65536 gso_max_segs 65535
 inet 192.0.2.1/24 brd 192.0.2.255 scope global noprefixroute vlan10
 valid_lft forever preferred_lft forever
 inet6 2001:db8:1::1/32 scope global noprefixroute
 valid_lft forever preferred_lft forever
 inet6 fe80::8dd7:9030:6f8e:89e6/64 scope link noprefixroute
 valid_lft forever preferred_lft forever
```

## 5.4. 使用 NM-CONNECTION-EDITOR 配置 VLAN 标记

您可以使用 **nm-connection-editor** 应用程序在图形界面中配置 Virtual Local Area Network (VLAN) 标记。

### 先决条件

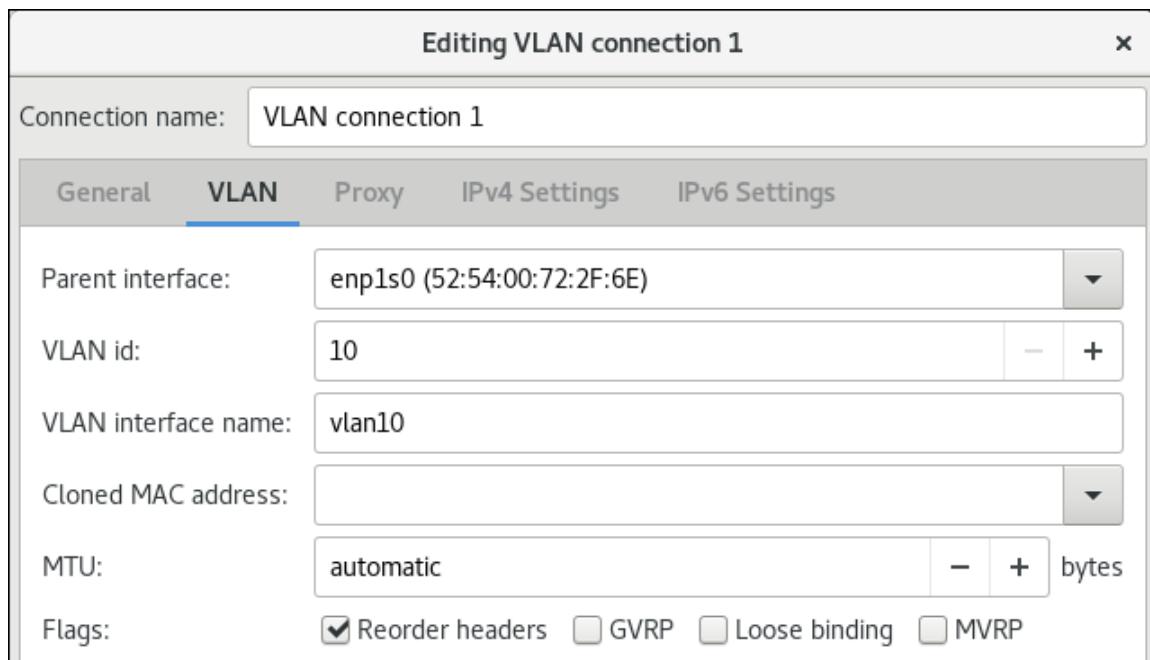
- 您计划用作虚拟 VLAN 接口的父接口支持 VLAN 标签。
- 如果您在绑定接口之上配置 VLAN：
  - 绑定的端口是上线的。
  - 这个绑定没有使用 **fail\_over\_mac=follow** 选项进行配置。VLAN 虚拟设备无法更改其 MAC 地址以匹配父设备的新 MAC 地址。在这种情况下，流量仍会与不正确的源 MAC 地址一同发送。
- 主机已连接，以支持 VLAN 标签。详情请查看您的交换机文档。

### 步骤

1. 打开一个终端，输入 **nm-connection-editor**：

```
$ nm-connection-editor
```

2. 点击 **+** 按钮来添加一个新的连接。
3. 选择 **VLAN** 连接类型，然后单击 **Create**。
4. 在 **VLAN** 选项卡中：
  - a. 选择上级接口。
  - b. 选择 VLAN ID。请注意，VLAN 必须在 0 到 4094 范围内。
  - c. 默认情况下，VLAN 连接会继承上级接口的最大传输单元（MTU）。另外，还可设置不同的 MTU 值。
  - d. 可选：设置 VLAN 接口的名称以及其它特定于 VLAN 的选项。



5. 在 IPv4 Settings 和 IPv6 Settings 标签页中配置 IP 地址设置：

- 如果您计划将此网桥设备用作其它设备的端口，请将 Method 字段设置为 Disabled。
- 要使用 DHCP，请将 Method 字段保留为默认值 Automatic (DHCP)。
- 要使用静态 IP 设置，请将 Method 字段设置为 Manual，并相应地填写字段：

| Editing VLAN connection 1 |         | Editing VLAN connection 1 |               |
|---------------------------|---------|---------------------------|---------------|
| General                   | VLAN    | Proxy                     | IPv4 Settings |
| Method: Manual            |         | Method: Manual            |               |
| <b>Addresses</b>          |         | <b>Addresses</b>          |               |
| Address                   | Netmask | Gateway                   | Prefix        |
| 192.0.2.1                 | 24      | 192.0.2.254               | 2001:db8:1::1 |
| DNS servers:              |         | DNS servers:              |               |
| 192.0.2.253               |         | 2001:db8:1::ffffd         |               |

6. 点 Save。

7. 关闭 nm-connection-editor。

### 验证

1. 验证设置：

```
ip -d addr show wlan10
4:vlan10@enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
state UP group default qlen 1000
 link/ether 52:54:00:d5:e0:fb brd ff:ff:ff:ff:ff:ff promiscuity 0
 vlan protocol 802.1Q id 10 <REORDER_HDR> numtxqueues 1 numrxqueues 1
 gso_max_size 65536 gso_max_segs 65535
 inet 192.0.2.1/24 brd 192.0.2.255 scope global noprefixroute wlan10
 valid_lft forever preferred_lft forever
 inet6 2001:db8:1::1/32 scope global noprefixroute
 valid_lft forever preferred_lft forever
 inet6 fe80::8dd7:9030:6f8e:89e6/64 scope link noprefixroute
 valid_lft forever preferred_lft forever
```

## 其他资源

- 配置 NetworkManager 以避免使用特定配置集提供默认网关

## 5.5. 使用 NMSTATECTL 配置 VLAN 标记

使用 **nmstatectl** 工具通过 Nmstate API 配置 Virtual Local Area Network VLAN。Nmstate API 确保设置配置后结果与配置文件匹配。如果有任何失败，**nmstatectl** 会自动回滚更改以避免系统处于不正确的状态。

根据您的环境，相应地调整 YAML 文件。例如，要使用与 VLAN 中以太网适配器不同的设备，请调整您在 VLAN 中使用的端口的 **base-iface** 属性和 **type** 属性。

### 先决条件

- 要将以太网设备用作 VLAN 中的端口，必须在服务器中安装物理或者虚拟以太网设备。
- nmstate** 软件包已安装。

### 步骤

- 创建一个包含以下内容的 YAML 文件，如 `~/create-vlan.yml`：

```

interfaces:
- name: vlan10
 type: vlan
 state: up
 ipv4:
 enabled: true
 address:
 - ip: 192.0.2.1
 prefix-length: 24
 dhcp: false
 ipv6:
 enabled: true
 address:
 - ip: 2001:db8:1::1
 prefix-length: 64
 autoconf: false
 dhcp: false
 vlan:
 base-iface: enp1s0
 id: 10
- name: enp1s0
 type: ethernet
 state: up

routes:
 config:
 - destination: 0.0.0.0/0
 next-hop-address: 192.0.2.254
 next-hop-interface: vlan10
```

```

- destination: ::/0
 next-hop-address: 2001:db8:1::fffe
 next-hop-interface: vlan10

dns-resolver:
 config:
 search:
 - example.com
 server:
 - 192.0.2.200
 - 2001:db8:1::ffbb

```

这些设置定义一个 ID 为 10 的 VLAN，它使用 **enp1s0** 设备。作为子设备，VLAN 连接有以下设置：

- 静态 IPv4 地址 - **192.0.2.1**，子网掩码为 /24
- 静态 IPv6 地址 - **2001:db8:1::1**，子网掩码为 /64
- IPv4 默认网关 - **192.0.2.254**
- IPv6 默认网关 - **2001:db8:1::fffe**
- IPv4 DNS 服务器 - **192.0.2.200**
- IPv6 DNS 服务器 - **2001:db8:1::ffbb**
- DNS 搜索域 - **example.com**

## 2. 将设置应用到系统：

```
nmstatectl apply ~/create-vlan.yml
```

## 验证

### 1. 显示设备和连接的状态：

```
nmcli device status
DEVICE TYPE STATE CONNECTION
vlan10 wlan connected wlan10
```

### 2. 显示连接配置集的所有设置：

```
nmcli connection show wlan10
connection.id: wlan10
connection.uuid: 1722970f-788e-4f81-bd7d-a86bf21c9df5
connection.stable-id: --
connection.type: wlan
connection.interface-name: wlan10
...
```

### 3. 以 YAML 格式显示连接设置：

```
nmstatectl show wlan10
```

## 其他资源

- 您系统上的 **nmstatectl (8)** 手册页
- **/usr/share/doc/nmstate/examples/** 目录

## 5.6. 使用 **NETWORK RHEL** 系统角色配置 VLAN 标记

如果您的网络使用虚拟局域网(VLAN)将网络流量分隔到逻辑网络中, 请创建一个 NetworkManager 连接配置文件来配置 VLAN 标记。通过使用 Ansible 和 **network** RHEL 系统角色, 您可以自动化此过程, 并在 playbook 中定义的主机上远程配置连接配置文件。

您可以使用 **network** RHEL 系统角色配置 VLAN 标记, 如果 VLAN 的父设备的连接配置文件不存在, 角色也可以创建它。



### 注意

如果 VLAN 设备需要 IP 地址、默认网关和 DNS 设置, 请在 VLAN 设备上而不是在父设备上配置它们。

### 前提条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户对其具有 **sudo** 权限。

### 步骤

1. 创建一个包含以下内容的 playbook 文件, 如 **~/playbook.yml** :

```

- name: Configure the network
 hosts: managed-node-01.example.com
 tasks:
 - name: VLAN connection profile with Ethernet port
 ansible.builtin.include_role:
 name: redhat.rhel_system_roles.network
 vars:
 network_connections:
 # Ethernet profile
 - name: enp1s0
 type: ethernet
 interface_name: enp1s0
 autoconnect: yes
 state: up
 ip:
 dhcp4: no
 auto6: no

 # VLAN profile
 - name: enp1s0.10
 type: vlan
 vlan:

```

```

id: 10
ip:
 dhcp4: yes
 auto6: yes
parent: enp1s0
state: up

```

示例 playbook 中指定的设置包括以下内容：

#### **type: <profile\_type>**

设置要创建的配置文件的类型。示例 playbook 创建两个连接配置文件：一个用于父以太网设备，另一个用于 VLAN 设备。

#### **dhcp4: <value>**

如果设置为 **yes**，则启用来自 DHCP、PPP 或类似服务的自动 IPv4 地址分配。禁用父设备上的 IP 地址配置。

#### **auto6: <value>**

如果设置为 **yes**，则启用 IPv6 自动配置。在这种情况下，NetworkManager 默认使用路由器通告，如果路由器宣布 **managed** 标志，NetworkManager 会从 DHCPv6 服务器请求 IPv6 地址和前缀。禁用父设备上的 IP 地址配置。

#### **parent: <parent\_device>**

设置 VLAN 连接配置文件的父设备。在示例中，父设备是以太网接口。

有关 playbook 中使用的所有变量的详情，请查看控制节点上的 **/usr/share/ansible/roles/rhel-system-roles.network/README.md** 文件。

## 2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

## 3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

## 验证

- 验证 VLAN 设置：

```

ansible managed-node-01.example.com -m command -a 'ip -d addr show enp1s0.10'
managed-node-01.example.com | CHANGED | rc=0 >>
4: vlan10@enp1s0.10: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
noqueue state UP group default qlen 1000
link/ether 52:54:00:72:2f:6e brd ff:ff:ff:ff:ff:ff promiscuity 0
 vlan protocol 802.1Q id 10 <REORDER_HDR> numtxqueues 1 numrxqueues 1
 gso_max_size 65536 gso_max_segs 65535
...

```

## 其他资源

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** 文件

- **/usr/share/doc/rhel-system-roles/network/ 目录**

## 第 6 章 配置网络桥接

网络桥接是一个链路层设备，它可根据 MAC 地址列表转发网络间的流量。网桥通过侦听网络流量并了解连接到每个网络的主机来构建 MAC 地址表。例如，您可以使用 Red Hat Enterprise Linux 主机上的软件桥接模拟硬件桥接或在虚拟化环境中，将虚拟机(VM)集成到与主机相同的网络中。

桥接需要在桥接应该连接的每个网络中有一个网络设备。当您配置网桥时，网桥被称为 **controller**，其使用的设备为 **ports**。

您可以在不同类型的设备中创建桥接，例如：

- 物理和虚拟以太网设备
- 网络绑定
- 网络团队 (team)
- VLAN 设备

由于 IEEE 802.11 标准指定在 Wi-Fi 中使用 3 个地址帧以便有效地使用随机时间，您无法通过 Ad-Hoc 或者 Infrastructure 模式中的 Wi-Fi 网络配置网桥。

### 6.1. 使用 **NMCLI** 配置网桥

要在命令行中配置网络桥接，请使用 **nmcli** 实用程序。

#### 先决条件

- 在服务器中安装两个或者两个以上物理或者虚拟网络设备。
- 要将以太网设备用作网桥的端口，必须在服务器中安装物理或者虚拟以太网设备。
- 要使用 team、bond 或 VLAN 设备作为网桥的端口，您可以在创建桥接时创建这些设备，或者预先创建它们，如：
  - 使用 [nmcli 配置网络团队](#)
  - 使用 [nmcli 配置网络绑定](#)
  - 使用 [nmcli 配置 VLAN 标记](#)

#### 步骤

1. 创建网桥接口：

```
nmcli connection add type bridge con-name bridge0 ifname bridge0
```

此命令创建名为 **bridge0** 的网桥。

2. 显示网络接口，并记录您要添加到网桥中的接口名称：

```
nmcli device status
DEVICE TYPE STATE CONNECTION
enp7s0 ethernet disconnected --
enp8s0 ethernet disconnected --
```

```
bond0 bond connected bond0
bond1 bond connected bond1
...

```

在本例中：

- 没有配置 **enp7s0** 和 **enp8s0**。要将这些设备用作端口，请在下一步中添加连接配置集。
  - **bond0** 和 **bond1** 已有连接配置文件。要将这些设备用作端口，请在下一步中修改其配置集。
3. 将接口分配给网桥。
    - a. 如果没有配置您要分配给网桥的接口，为其创建新的连接配置集：

```
nmcli connection add type ethernet slave-type bridge con-name bridge0-port1
 ifname enp7s0 master bridge0
nmcli connection add type ethernet slave-type bridge con-name bridge0-port2
 ifname enp8s0 master bridge0
```

这些命令为 **enp7s0** 和 **enp8s0** 创建配置文件，并将它们添加到 **bridge0** 连接中。

- b. 如果要为网桥分配现有连接配置集：

- i. 将这些连接的 **master** 参数设置为 **bridge0**：

```
nmcli connection modify bond0 master bridge0
nmcli connection modify bond1 master bridge0
```

这些命令将名为 **bond0** 和 **bond1** 的现有连接配置文件分配给 **bridge0** 连接。

- ii. 重新激活连接：

```
nmcli connection up bond0
nmcli connection up bond1
```

4. 配置 IPv4 设置：

- 要为 **bridge0** 连接设置静态 IPv4 地址、网络掩码、默认网关和 DNS 服务器设置，请输入：

```
nmcli connection modify bridge0 ipv4.addresses '192.0.2.1/24' ipv4.gateway
 '192.0.2.254' ipv4.dns '192.0.2.253' ipv4.dns-search 'example.com' ipv4.method
 manual
```

- 要使用 DHCP，不需要任何操作。
- 如果您计划将此网桥设备用作其它设备的端口，则不需要任何操作。

5. 配置 IPv6 设置：

- 要为 **bridge0** 连接设置静态 IPv6 地址、网络掩码、默认网关和 DNS 服务器设置，请输入：

```
nmcli connection modify bridge0 ipv6.addresses '2001:db8:1::1/64' ipv6.gateway
 '2001:db8:1::fffe' ipv6.dns '2001:db8:1::ffff' ipv6.dns-search 'example.com'
 ipv6.method manual
```

- 要使用无状态地址自动配置(SLAAC)，不需要采取任何操作。
  - 如果您计划将此网桥设备用作其它设备的端口，则不需要任何操作。
6. 可选：配置网桥的其他属性。例如，要将 **bridge0** 的生成树协议(STP)优先级设为 **16384**，请输入：

```
nmcli connection modify bridge0 bridge.priority '16384'
```

默认情况下启用 STP。

7. 激活连接：

```
nmcli connection up bridge0
```

8. 验证端口是否已连接，并且 **CONNECTION** 列是否显示端口的连接名称：

```
nmcli device
DEVICE TYPE STATE CONNECTION
...
enp7s0 ethernet connected bridge0-port1
enp8s0 ethernet connected bridge0-port2
```

当您激活连接的任何端口时，NetworkManager 也会激活网桥，但不会激活它的其它端口。您可以配置 Red Hat Enterprise Linux 在启用桥接时自动启用所有端口：

- a. 启用网桥连接的 **connection.autoconnect-slaves** 参数：

```
nmcli connection modify bridge0 connection.autoconnect-slaves 1
```

- b. 重新激活桥接：

```
nmcli connection up bridge0
```

## 验证

- 使用 **ip** 工具来显示作为特定网桥端口的以太网设备的链接状态：

```
ip link show master bridge0
3: enp7s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master
 bridge0 state UP mode DEFAULT group default qlen 1000
 link/ether 52:54:00:62:61:0e brd ff:ff:ff:ff:ff:ff
4: enp8s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master
 bridge0 state UP mode DEFAULT group default qlen 1000
 link/ether 52:54:00:9e:f1:ce brd ff:ff:ff:ff:ff:ff
```

- 使用 **bridge** 工具来显示作为任意网桥设备端口的以太网设备状态：

```
bridge link show
3: enp7s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master bridge0 state
 forwarding priority 32 cost 100
4: enp8s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master bridge0 state
 listening priority 32 cost 100
```

```

5: enp9s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master bridge1 state
forwarding priority 32 cost 100
6: enp11s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master bridge1 state
blocking priority 32 cost 100
...

```

要显示特定以太网设备的状态，请使用 **bridge link show dev <ethernet\_device\_name>** 命令。

## 其他资源

- 您系统上的 **bridge (8)** 和 **nm-settings (5)** 手册页
- 重启 NetworkManager 服务后 NetworkManager 会复制连接 (红帽知识库)
- 如何配置具有 VLAN 信息的网桥？ (红帽知识库)

## 6.2. 使用 RHEL WEB 控制台配置网桥

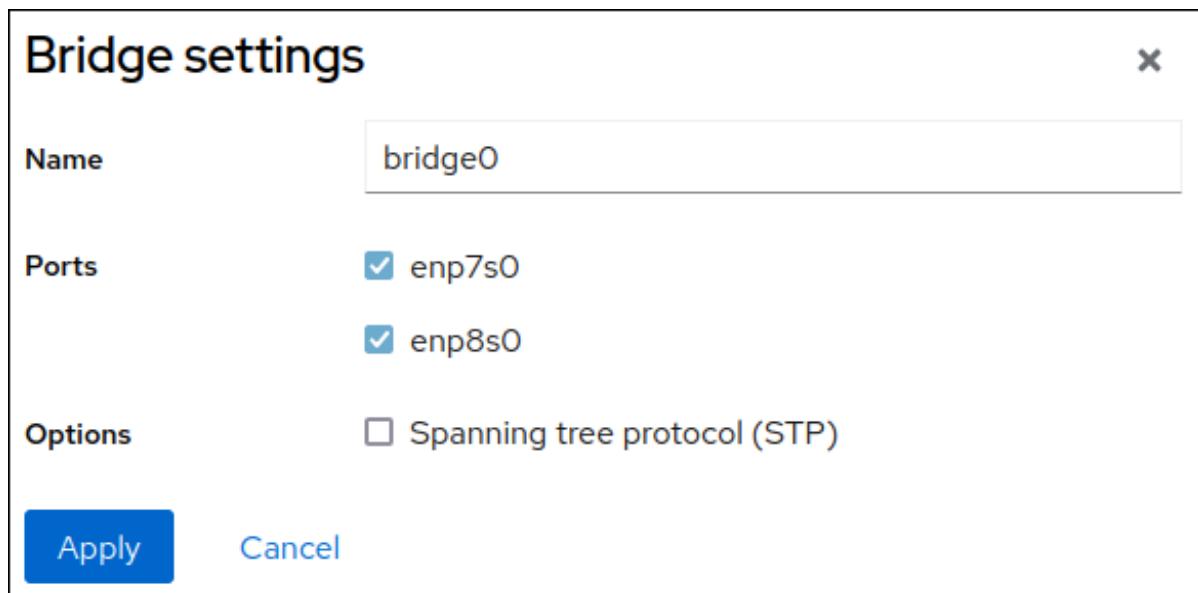
如果您希望通过基于 Web 浏览器的界面管理网络设置，请使用 RHEL web 控制台来配置网桥。

### 前提条件

- 在服务器中安装两个或者两个以上物理或者虚拟网络设备。
- 要将以太网设备用作网桥的端口，必须在服务器中安装物理或者虚拟以太网设备。
- 要使用 team、bond 或 VLAN 设备作为网桥的端口，您可以在创建桥接时创建这些设备，或者预先创建它们，如：
  - 使用 RHEL web 控制台配置网络团队
  - 使用 RHEL web 控制台配置网络绑定
  - 使用 RHEL web 控制台配置 VLAN 标记
- 您已安装了 RHEL 8 web 控制台。
- 您已启用了 cockpit 服务。
- 您的用户帐户被允许登录到 web 控制台。  
具体步骤请参阅[安装并启用 Web 控制台](#)。

### 流程

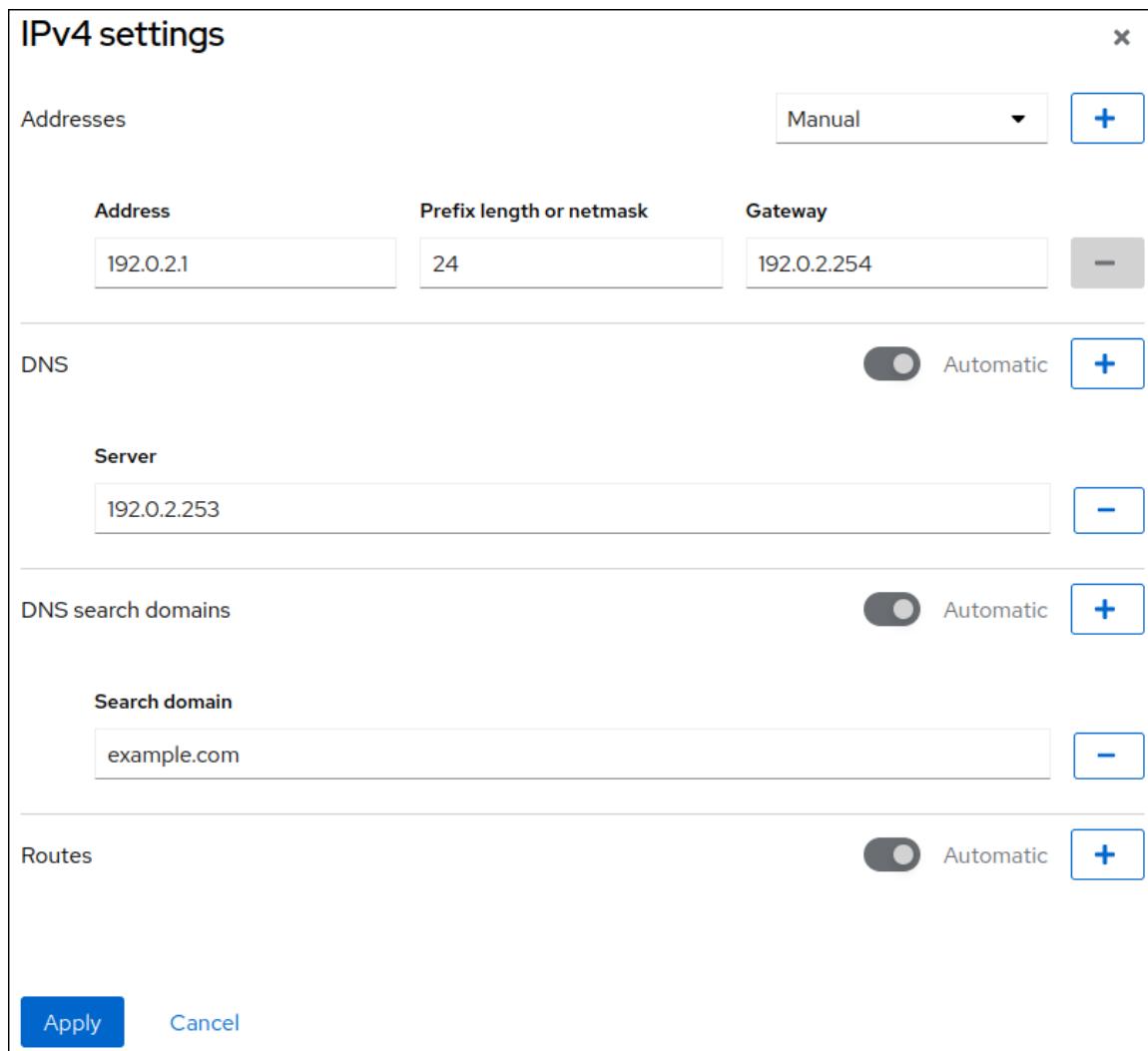
1. 登录到 RHEL 8 web 控制台。  
详情请参阅[登录到 web 控制台](#)。
2. 在屏幕左侧的导航中选择 **Networking** 选项卡。
3. 在 **Interfaces** 部分点 **Add bridge**。
4. 输入您要创建的网桥设备名称。
5. 选择应该是网桥端口的接口。
6. 可选：启用 **Spanning tree 协议(STP)** 功能，以避免桥循环和广播辐射。



7. 点应用。

8. 默认情况下，网桥使用动态 IP 地址。如果要设置静态 IP 地址：

- a. 在 **Interfaces** 部分，点网桥的名称。
- b. 点您要配置的协议旁的 **Edit**。
- c. 选择 **Addresses** 旁的 **Manual**，并输入 IP 地址、前缀和默认网关。
- d. 在 **DNS** 部分，点 **+** 按钮，并输入 DNS 服务器的 IP 地址。重复此步骤来设置多个 DNS 服务器。
- e. 在 **DNS search domains** 部分中，点 **+** 按钮并输入搜索域。
- f. 如果接口需要静态路由，请在 **Routes** 部分配置它们。



g. 点应用

### 验证

- 在屏幕左侧的导航中选择 Networking 选项卡，并检查接口上是否有传入和传出流量：

| Interfaces |              | <a href="#">Add bond</a> | <a href="#">Add team</a> | <a href="#">Add bridge</a> | <a href="#">Add VLAN</a> |
|------------|--------------|--------------------------|--------------------------|----------------------------|--------------------------|
| Name       | IP address   | Sending                  | Receiving                |                            |                          |
| bridge0    | 192.0.2.1/24 | 1.11 Mbps                | 61.2 Mbps                |                            |                          |

## 6.3. 使用 nmtui 配置网桥

**nmtui** 应用程序为 NetworkManager 提供了一个基于文本的用户界面。您可以使用 **nmtui** 在没有图形界面的主机上配置网桥。



## 注意

在 **nmtui** 中：

- 使用光标键导航。
- 选择一个按钮并按 **Enter** 键。
- 使用 **空格** 选择和清除复选框。
- 要返回上一个屏幕，请使用 **ESC**。

## 先决条件

- 在服务器中安装两个或者两个以上物理或者虚拟网络设备。
- 要将以太网设备用作网桥的端口，必须在服务器中安装物理或者虚拟以太网设备。

## 流程

1. 如果您不知道您要在其上配置网桥的网络设备名称，请显示可用的设备：

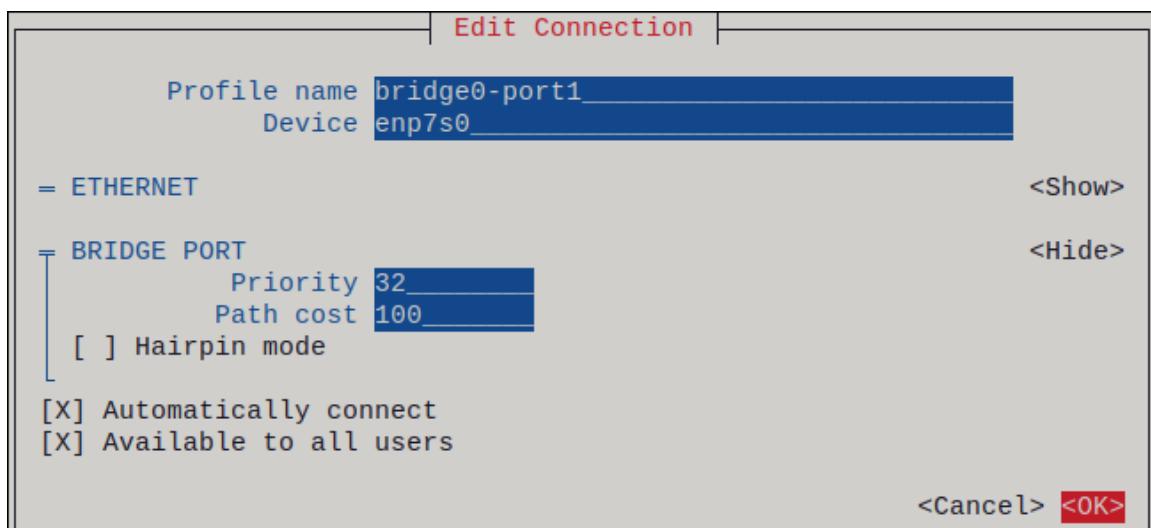
```
nmcli device status
DEVICE TYPE STATE CONNECTION
enp7s0 ethernet unavailable --
enp8s0 ethernet unavailable --
...
```

2. 启动 **nmtui**：

```
nmtui
```

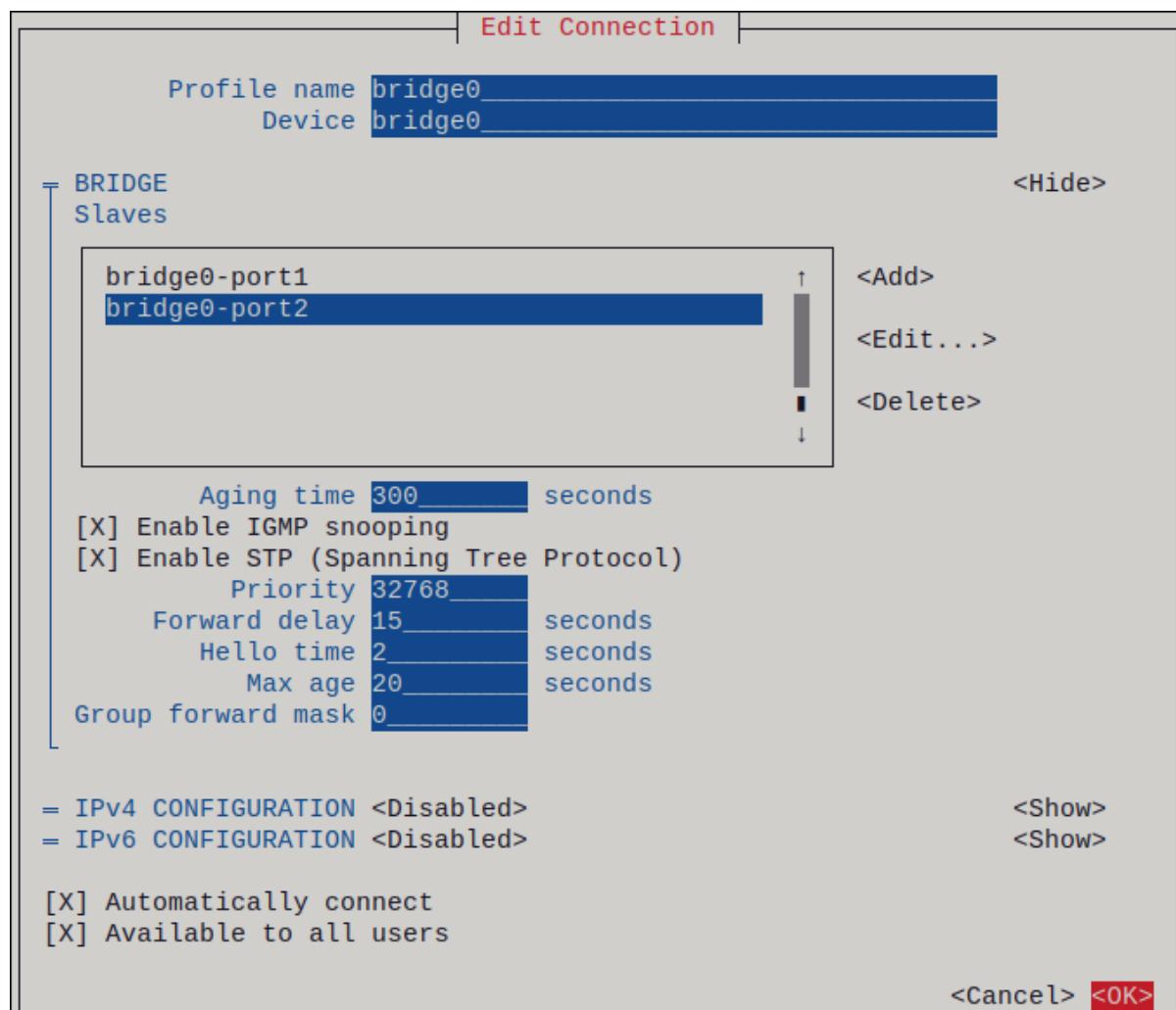
3. 选择 **Edit a connection**，点 **Enter**。
4. 按 **Add**。
5. 从网络类型列表中选择 **Bridge**，然后按 **Enter**。  
在有多个配置文件的主机上，有意义的名称可以更容易识别配置文件的用途。
6. 可选：为要创建的 NetworkManager 配置集输入一个名称。  
在有多个配置文件的主机上，有意义的名称可以更容易识别配置文件的用途。
7. 在 **Device** 字段中输入要创建的网桥设备名称。
8. 将端口添加到要创建的网桥中：
  - a. 按 **Slaves** 列表旁边的 **Add**。
  - b. 选择您要作为端口添加到网桥的接口类型，例如 **Ethernet**。
  - c. 可选：为这个网桥端口输入要创建的 NetworkManager 配置文件的名称。
  - d. 在 **Device** 字段中输入端口的设备名称。
  - e. 按 **OK** 返回到网桥设置窗口。

图 6.1. 将以太网设备作为端口添加到网桥



- f. 重复这些步骤，来向网桥添加更多的端口。
9. 根据您的环境，相应地在 **IPv4 configuration** 和 **IPv6 configuration** 区中配置 IP 地址。为此，请按这些区域旁边的按钮，并选择：
- **Disabled**，如果网桥不需要 IP 地址。
  - **Automatic**，如果 DHCP 服务器或无状态地址自动配置(SLAAC)动态将 IP 地址分配给网桥。
  - **Manual**，如果网络需要静态 IP 地址设置。在这种情况下，您必须填写更多字段：
    - i. 按您要配置的协议旁边的 **Show** 来显示其他字段。
    - ii. 按 **Addresses** 旁边的 **Add**，并输入无类别域间路由(CIDR)格式的 IP 地址和子网掩码。如果没有指定子网掩码，NetworkManager 会为 IPv4 地址设置 /32 子网掩码，并为 IPv6 地址设置 /64。
    - iii. 输入默认网关的地址。
    - iv. 按 **DNS servers** 旁边的 **Add**，并输入 DNS 服务器地址。
    - v. 按 **Search domains** 旁边的 **Add**，并输入 DNS 搜索域。

图 6.2. 没有 IP 地址设置的网桥连接的示例



10. 按 **OK** 创建并自动激活新连接。
11. 按 **Back** 返回到主菜单。
12. 选择 **Quit**, 然后按 **Enter** 键关闭 **nmtui** 应用程序。

## 验证

1. 使用 **ip** 工具来显示作为特定网桥端口的以太网设备的链接状态：

```
ip link show master bridge0
3: enp7s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master
 bridge0 state UP mode DEFAULT group default qlen 1000
 link/ether 52:54:00:62:61:0e brd ff:ff:ff:ff:ff:ff
4: enp8s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master
 bridge0 state UP mode DEFAULT group default qlen 1000
 link/ether 52:54:00:9e:f1:ce brd ff:ff:ff:ff:ff:ff
```

2. 使用 **bridge** 工具来显示作为任意网桥设备端口的以太网设备状态：

```
bridge link show
3: enp7s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master bridge0 state
forwarding priority 32 cost 100
```

```
4: enp8s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master bridge0 state listening priority 32 cost 100
```

...

要显示特定以太网设备的状态，请使用 **bridge link show dev <ethernet\_device\_name>** 命令。

## 6.4. 使用 NM-CONNECTION-EDITOR 配置网桥

如果使用带有图形界面的 Red Hat Enterprise Linux，您可以使用 **nm-connection-editor** 应用程序配置网桥。

请注意，**nm-connection-editor** 只能向网桥添加新端口。要使用现有连接配置文件作为端口，请使用 **nmcli** 工具创建网桥，如 [使用 nmcli 配置网桥](#) 中所述。

### 先决条件

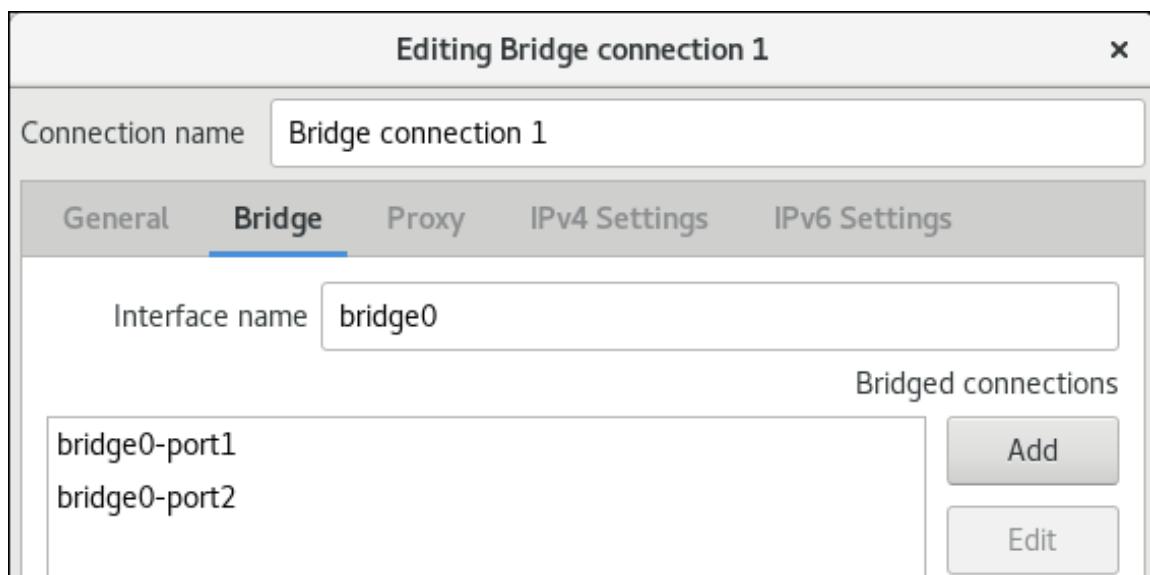
- 在服务器中安装两个或者两个以上物理或者虚拟网络设备。
- 要将以太网设备用作网桥的端口，必须在服务器中安装物理或者虚拟以太网设备。
- 要使用 team、bond 或 VLAN 设备作为网桥的端口，请确保这些设备还没有配置。

### 流程

- 打开一个终端，输入 **nm-connection-editor**：

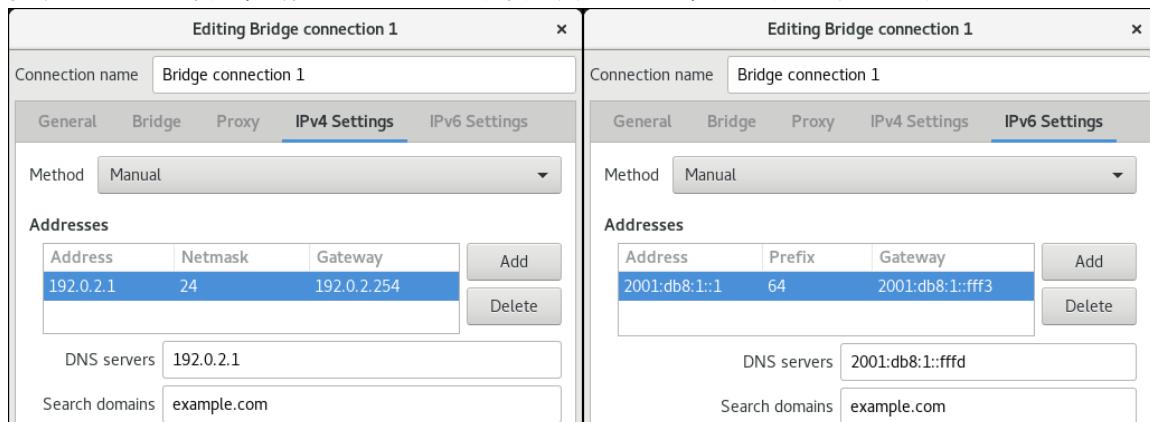
```
$ nm-connection-editor
```

- 点击 **+** 按钮来添加一个新的连接。
- 选择 **Bridge** 连接类型，然后单击 **Create**。
- 在 **Bridge** 选项卡中：
  - 可选：在 **Interface name** 字段中设置网桥接口的名称。
  - 点 **Add** 按钮为网络接口创建新连接配置集，并将配置集作为端口添加到网桥。
    - 选择接口的连接类型。例如，为有线连接选择 **Ethernet**。
    - 可选：为端口设备设置连接名称。
    - 如果您为以太网设备创建连接配置文件，请打开 **Ethernet** 选项卡，然后在 **Device** 字段中选择您要作为端口添加到网桥的网络接口。如果您选择了不同的设备类型，请相应地进行配置。
  - 点 **Save**。
  - 对您要添加到桥接的每个接口重复前面的步骤。



5. 可选：配置其他网桥设置，如生成树协议（STP）选项。
6. 在 IPv4 Settings 和 IPv6 Settings 标签页中配置 IP 地址设置：

- 如果您计划将此网桥设备用作其它设备的端口，请将 Method 字段设置为 Disabled。
- 要使用 DHCP，请将 Method 字段保留为默认值 Automatic (DHCP)。
- 要使用静态 IP 设置，请将 Method 字段设置为 Manual，并相应地填写字段：



7. 点击 Save。
8. 关闭 nm-connection-editor。

## 验证

- 使用 ip 工具来显示作为特定网桥端口的以太网设备的链接状态。

```
ip link show master bridge0
3: enp7s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master
 bridge0 state UP mode DEFAULT group default qlen 1000
 link/ether 52:54:00:62:61:0e brd ff:ff:ff:ff:ff:ff
4: enp8s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master
 bridge0 state UP mode DEFAULT group default qlen 1000
 link/ether 52:54:00:9e:f1:ce brd ff:ff:ff:ff:ff:ff
```

- 使用 bridge 工具来显示作为任意网桥设备中端口的以太网设备状态：

```
bridge link show
3: enp7s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master bridge0 state
forwarding priority 32 cost 100
4: enp8s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master bridge0 state
listening priority 32 cost 100
5: enp9s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master bridge1 state
forwarding priority 32 cost 100
6: enp11s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master bridge1 state
blocking priority 32 cost 100
...
```

要显示特定以太网设备的状态，请使用 **bridge link show dev ethernet\_device\_name** 命令。

## 其他资源

- [使用 nm-connection-editor 配置网络绑定](#)
- [使用 nm-connection-editor 配置网络团队](#)
- [使用 nm-connection-editor 配置 VLAN 标记](#)
- [配置 NetworkManager 以避免使用特定配置集提供默认网关](#)
- [如何配置具有 VLAN 信息的网桥？（红帽知识库）](#)

## 6.5. 使用 NMSTATECTL 配置网桥

使用 **nmstatectl** 工具通过 Nmstate API 配置网桥。Nmstate API 确保设置配置后结果与配置文件匹配。如果有任何失败，**nmstatectl** 会自动回滚更改以避免系统处于不正确的状态。

根据您的环境，相应地调整 YAML 文件。例如，要使用与网桥中以太网适配器不同的设备，请调整您在网桥中使用的端口的 **base-iface** 属性和 **type** 属性。

### 先决条件

- 在服务器中安装两个或者两个以上物理或者虚拟网络设备。
- 要将以太网设备用作网桥中的端口，必须在服务器中安装物理或者虚拟以太网设备。
- 要使用团队、绑定或 VLAN 设备作为网桥中的端口，请在 **port** 列表中设置接口名称，并定义对应的接口。
- **nmstate** 软件包已安装。

### 流程

1. 创建一个包含以下内容的 YAML 文件，如 `~/create-bridge.yml`：

```

interfaces:
- name: bridge0
 type: linux-bridge
 state: up
 ipv4:
```

```

enabled: true
address:
- ip: 192.0.2.1
 prefix-length: 24
dhcp: false
ipv6:
 enabled: true
 address:
 - ip: 2001:db8:1::1
 prefix-length: 64
 autoconf: false
 dhcp: false
bridge:
 options:
 stp:
 enabled: true
 port:
 - name: enp1s0
 - name: enp7s0
- name: enp1s0
type: ethernet
state: up
- name: enp7s0
type: ethernet
state: up

routes:
 config:
 - destination: 0.0.0.0/0
 next-hop-address: 192.0.2.254
 next-hop-interface: bridge0
 - destination: ::/0
 next-hop-address: 2001:db8:1::fffe
 next-hop-interface: bridge0
dns-resolver:
 config:
 search:
 - example.com
 server:
 - 192.0.2.200
 - 2001:db8:1::ffbb

```

这些设置使用以下设置定义一个网桥：

- 网桥中的网络接口 : **enp1s0** 和 **enp7s0**
- Spanning Tree Protocol (STP): 启用
- 静态 IPv4 地址 : **192.0.2.1**, 子网掩码为 /**24**
- 静态 IPv6 地址 : **2001:db8:1::1**, 子网掩码为 /**64**
- IPv4 默认网关 : **192.0.2.254**
- IPv6 默认网关 : **2001:db8:1::fffe**

- IPv4 DNS 服务器 : **192.0.2.200**
- IPv6 DNS 服务器 : **2001:db8:1::ffbb**
- DNS 搜索域 : **example.com**

2. 将设置应用到系统 :

```
nmstatectl apply ~/create-bridge.yml
```

验证

1. 显示设备和连接的状态 :

```
nmcli device status
DEVICE TYPE STATE CONNECTION
bridge0 bridge connected bridge0
```

2. 显示连接配置集的所有设置 :

```
nmcli connection show bridge0
connection.id: bridge0_
connection.uuid: e2cc9206-75a2-4622-89cf-1252926060a9
connection.stable-id: --
connection.type: bridge
connection.interface-name: bridge0
...
```

3. 以 YAML 格式显示连接设置 :

```
nmstatectl show bridge0
```

## 其他资源

- 您系统上的 **nmstatectl (8)** 手册页
- **/usr/share/doc/nmstate/examples/** 目录
- [如何配置具有 VLAN 信息的网桥？](#) (红帽知识库)

## 6.6. 使用 NETWORK RHEL 系统角色配置网桥

您可以通过创建网桥，在 Open Systems Interconnection (OSI) 模型的第 2 层上连接多个网络。要配置网桥，请在 NetworkManager 中创建一个连接配置文件。通过使用 Ansible 和 **network** RHEL 系统角色，您可以自动化此过程，并在 playbook 中定义的主机上远程配置连接配置文件。

您可以使用 **network** RHEL 系统角色配置网桥，如果网桥的父设备的连接配置文件不存在，则角色也可以创建它。



### 注意

如果要为网桥分配 IP 地址、网关和 DNS 设置，请在网桥上，而不是在其端口上配置它们。

## 先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 您用于连接到受管节点的帐户对它们具有 **sudo** 权限。
- 在服务器中安装两个或者两个以上物理或者虚拟网络设备。

## 流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```

- name: Configure the network
 hosts: managed-node-01.example.com
 tasks:
 - name: Bridge connection profile with two Ethernet ports
 ansible.builtin.include_role:
 name: redhat.rhel_system_roles.network
 vars:
 network_connections:
 # Bridge profile
 - name: bridge0
 type: bridge
 interface_name: bridge0
 ip:
 dhcp4: yes
 auto6: yes
 state: up

 # Port profile for the 1st Ethernet device
 - name: bridge0-port1
 interface_name: enp7s0
 type: ethernet
 controller: bridge0
 port_type: bridge
 state: up

 # Port profile for the 2nd Ethernet device
 - name: bridge0-port2
 interface_name: enp8s0
 type: ethernet
 controller: bridge0
 port_type: bridge
 state: up
```

示例 playbook 中指定的设置包括如下：

**type: <profile\_type>**

设置要创建的配置文件的类型。示例 playbook 创建三个连接配置文件：一个用于网桥，两个用于以太网设备。

**dhcp4: yes**

启用来自 DHCP、PPP 或类似服务的自动 IPv4 地址分配。

**auto6: yes**

启用 IPv6 自动配置。默认情况下，NetworkManager 使用路由器公告。如果路由器宣布 **managed** 标记，则 NetworkManager 会从 DHCPv6 服务器请求 IPv6 地址和前缀。

有关 playbook 中使用的所有变量的详情，请查看控制节点上的 **/usr/share/ansible/roles/rhel-system-roles.network/README.md** 文件。

- 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不能防止错误的、但有效的配置。

- 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

## 验证

- 显示作为特定网桥端口的以太网设备的链接状态：

```
ansible managed-node-01.example.com -m command -a 'ip link show master bridge0'
managed-node-01.example.com | CHANGED | rc=0 >>
3: enp7s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master
bridge0 state UP mode DEFAULT group default qlen 1000
 link/ether 52:54:00:62:61:0e brd ff:ff:ff:ff:ff:ff
4: enp8s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master
bridge0 state UP mode DEFAULT group default qlen 1000
 link/ether 52:54:00:9e:f1:ce brd ff:ff:ff:ff:ff:ff
```

- 显示作为任意网桥设备的端口的以太网设备状态：

```
ansible managed-node-01.example.com -m command -a 'bridge link show'
managed-node-01.example.com | CHANGED | rc=0 >>
3: enp7s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master bridge0 state
forwarding priority 32 cost 100
4: enp8s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 master bridge0 state
listening priority 32 cost 100
```

## 其他资源

- /usr/share/ansible/roles/rhel-system-roles.network/README.md** 文件
- /usr/share/doc/rhel-system-roles/network/** 目录

# 第 7 章 设置 IPSEC VPN

虚拟专用网络(VPN)是一种通过互联网连接到本地网络的方法。**Libreswan** 提供的 **IPsec** 是创建 VPN 的首选方法。**libreswan** 是 VPN 的用户空间 **IPsec** 实现。VPN 通过在中间网络（如互联网）设置隧道来启用 LAN 和另一个远程 LAN 之间的通信。为了安全起见，VPN 隧道总是使用认证和加密。对于加密操作，**Libreswan** 使用 **NSS** 库。

## 7.1. LIBRESWAN 作为 IPSEC VPN 的实现

在 RHEL 中，您可以使用 Libreswan 应用程序支持的 IPsec 协议配置虚拟专用网络(VPN)。Libreswan 是 Openswan 应用程序的延续，Openswan 文档中的许多示例可以通过 Libreswan 交换。

VPN 的 IPsec 协议使用互联网密钥交换(IKE)协议进行配置。术语 IPsec 和 IKE 可互换使用。IPsec VPN 也称为 IKE VPN、IKEv2 VPN、XAUTH VPN、Cisco VPN 或 IKE/IPsec VPN。IPsec VPN 变体，它使用 Level 2 Tunneling Protocol(L2TP)，被称为 L2TP/IPsec VPN，它需要 **optional** 软件仓库提供的 **xlt2tpd** 软件包。

**libreswan** 是一个开源用户空间 IKE 实现。IKE v1 和 v2 作为用户级别的守护进程实现。IKE 协议也加密。IPsec 协议由 Linux 内核实现，Libreswan 配置内核以添加和删除 VPN 隧道配置。

IKE 协议使用 UDP 端口 500 和 4500。IPsec 协议由两个协议组成：

- 封装安全性 Payload(ESP)，其协议号为 50。
- 经过身份验证的标头(AH)，其协议号为 51。

不建议使用 AH 协议。建议将 AH 用户迁移到使用 null 加密的 ESP。

IPsec 协议提供两种操作模式：

- 隧道模式（默认）
- 传输模式

您可以用没有 IKE 的 IPsec 来配置内核。这称为 手动键控。您还可以使用 **ip xfrm** 命令来配置手动密钥，但为了安全起见，强烈建议您不要这样做。Libreswan 使用 Netlink 接口与 Linux 内核进行通信。内核执行数据包加密和解密。

Libreswan 使用网络安全服务 (NSS) 加密库。NSS 认证用于 [联邦信息处理标准\(FIPS\)](#) 出版物 140-2。



### 重要

IKE/IPsec VPN（由 Libreswan 和 Linux 内核实现）是 RHEL 中推荐的唯一 VPN 技术。在不了解这样做风险的情况下不要使用任何其他 VPN 技术。

在 RHEL 中，Libreswan 默认[遵循系统范围的加密策略](#)。这样可确保 Libreswan 将当前威胁模型包括 (IKEv2) 的安全设置用作默认协议。如需更多信息，请参阅 [使用系统范围的加密策略](#)。

Libreswan 没有使用术语“源（source）”和“目的地（destination）”或“服务器（server）”和“客户端（client）”，因为 IKE/IPsec 使用对等（peer to peer）协议。相反，它使用术语“左”和“右”来指端点（主机）。这也允许您在大多数情况下在两个端点使用相同的配置。但是，管理员通常选择始终对本地主机使用“左”，对远程主机使用“右”。

**leftid** 和 **rightid** 选项充当身份验证过程中相应主机的标识。详情请查看 **ipsec.conf(5)** 手册页。

## 7.2. LIBRESWAN 中的身份验证方法

Libreswan 支持多种身份验证方法，每种方法适合不同的场景。

### 预共享密钥(PSK)

预共享密钥(PSK)是最简单的身份验证方法。出于安全考虑，请勿使用小于 64 个随机字符的 PSK。在 FIPS 模式中，PSK 必须符合最低强度要求，具体取决于所使用的完整性算法。您可以使用 **authby=secret** 连接来设置 PSK。

### 原始 RSA 密钥

原始 RSA 密钥通常用于静态主机到主机或子网到子网 IPsec 配置。每个主机都使用所有其他主机的公共 RSA 密钥手动配置，Libreswan 在每对主机之间建立 IPsec 隧道。对于大量主机，这个方法不能很好地扩展。

您可以使用 **ipsec newhostkey** 命令在主机上生成原始 RSA 密钥。您可以使用 **ipsec showhostkey** 命令列出生成的密钥。使用 CKA ID 密钥的连接配置需要 **leftrsasigkey=** 行。原始 RSA 密钥使用 **authby=rsasig** 连接选项。

### X.509 证书

X.509 证书通常用于大规模部署连接到通用 IPsec 网关的主机。中心 证书颁发机构(CA)为主机或用户签署 RSA 证书。此中央 CA 负责中继信任，包括单个主机或用户的撤销。

例如，您可以使用 **openssl** 命令和 NSS **certutil** 命令来生成 X.509 证书。因为 Libreswan 使用 **leftcert=** 配置选项中证书的昵称从 NSS 数据库读取用户证书，所以在创建证书时请提供昵称。

如果使用自定义 CA 证书，则必须将其导入到网络安全服务(NSS)数据库中。您可以使用 **ipsec import** 命令将 PKCS #12 格式的任何证书导入到 Libreswan NSS 数据库。



#### 警告

Libreswan 需要互联网密钥交换(IKE)对等 ID 作为每个对等证书的主题替代名称(SAN)，如 [RFC 4945 的 3.1 章节](#) 所述。通过设置 **require-id-on-certificate=no** connection 选项禁用此检查，可能会导致系统容易受到中间人攻击。

使用 **authby=rsasig** 连接选项，根据使用带 SHA-1 和 SHA-2 的 RSA 的 X.509 证书进行身份验证。您可以通过将 **authby=** 设为 **ecdsa**，使用 SHA-2，以及使用通过 **authby=rsa-sha2** 的基于 SHA-2 的身份验证的 RSA 概率签名方案(RSASSA-PSS)数字签名来进一步限制 ECDSA 数字签名。默认值为 **authby=rsasig,ecdsa**。

证书和 **authby=** 签名方法应匹配。这提高了互操作性，并在一个数字签名系统中保留身份验证。

### NULL 身份验证

NULL 身份验证用来在没有身份验证的情况下获得网状加密。它可防止被动攻击，但不能防止主动攻击。但是，因为 IKEv2 允许非对称身份验证方法，因此 NULL 身份验证也可用于互联网规模的机会主义 IPsec。在此模型中，客户端对服务器进行身份验证，但服务器不对客户端进行身份验证。此模型类似于使用 TLS 的安全网站。使用 **authby=null** 进行 NULL 身份验证。

### 保护量子计算机

除了上述身份验证方法外，您还可以使用 *Post-quantum Pre-shared Key (PPK)*方法来防止量子计算机可能的攻击。单个客户端或客户端组可以通过指定与带外配置的预共享密钥对应的 PPK ID 来使用它们自己的 PPK。

使用带有预共享密钥的 IKEv1 防止量子攻击者。重新设计 IKEv2 不会原生提供这种保护。Libreswan 提供使用 *Post-quantum Pre-shared Key (PPK)*来保护 IKEv2 连接免受量子攻击。

要启用可选的 PPK 支持，请在连接定义中添加 **ppk=yes**。如需要 PPK，请添加 **ppk=insist**。然后，可以给每个客户端一个带有 secret 值的 PPK ID，该 secret 值在带外进行通信（最好是量子安全的）。PPK 应该具有很强的随机性，而不是基于字典中的单词。PPK ID 和 PPK 数据保存在 **ipsec.secrets** 文件中，例如：

```
@west @east : PPKS "user1" "thestringsmeanttobearandomstr"
```

**PPKS** 选项指的是静态 PPK。这个实验性功能使用基于一次性平板的动态 PPK。在每个连接中，一次性平板的一个新部件用作 PPK。当使用时，文件中动态 PPK 的那部分被零覆盖，以防止重复使用。如果没有剩下一次性资源，连接会失败。详情请查看 **ipsec.secrets(5)** 手册页。



### 警告

动态 PPK 的实现是作为不受支持的技术预览提供的。请谨慎使用。

## 7.3. 安装 LIBRESWAN

在通过 Libreswan IPsec/IKE 实现设置 VPN 之前，您必须安装相应的软件包，启动 **ipsec** 服务，并在防火墙中允许服务。

### 先决条件

- **AppStream** 存储库已启用。

### 流程

1. 安装 **libreswan** 软件包：

```
yum install libreswan
```

2. 如果要重新安装 Libreswan，请删除其旧的数据库文件，并创建一个新的数据库：

```
systemctl stop ipsec
rm /etc/ipsec.d/*db
ipsec initnss
```

3. 启动 **ipsec** 服务，并启用该服务，以便其在引导时自动启动：

```
systemctl enable ipsec --now
```

4. 通过添加 **ipsec** 服务，将防火墙配置为允许 IKE、ESP 和 AH 协议的 500 和 4500/UDP 端口：

```
firewall-cmd --add-service="ipsec"
firewall-cmd --runtime-to-permanent
```

## 7.4. 创建主机到主机的 VPN

您可以使用原始 RSA 密钥身份验证将 Libreswan 配置为在两个称为 *left* 和 *right* 的主机之间创建主机到主机的 IPsec VPN。

### 先决条件

- Libreswan 已安装，并在每个节点上启动了 **ipsec** 服务。

### 流程

1. 在每台主机上生成原始 RSA 密钥对：

```
ipsec newhostkey
```

2. 上一步返回生成的密钥的 **ckaid**。在左主机上使用 **ckaid** 和以下命令，例如：

```
ipsec showhostkey --left --ckaid 2d3ea57b61c9419dfd6cf43a1eb6cb306c0e857d
```

上一步命令的输出生成了配置所需的 **leftrsasigkey=** 行。在第二台主机（右）上执行相同的操作：

```
ipsec showhostkey --right --ckaid a9e1f6ce9ecd3608c24e8f701318383f41798f03
```

3. 在 **/etc/ipsec.d/** 目录中，创建一个新的 **my\_host-to-host.conf** 文件。将上一步中 **ipsec showhostkey** 命令的输出中的 RSA 主机密钥写入新文件。例如：

```
conn mytunnel
 leftid=@west
 left=192.1.2.23
 lefrsasigkey=0sAQOrlo+hOafUZDICQmXFrje/oZm [...] W2n417C/4urYHQkCvulQ==
 rightid=@east
 right=192.1.2.45
 rightrsasigkey=0sAQO3fwC6nSSGgt64DWiYZzuHbc4 [...] D/v8t5YTQ==
 authby=rsasig
```

4. 导入密钥后，重启 **ipsec** 服务：

```
systemctl restart ipsec
```

5. 加载连接：

```
ipsec auto --add mytunnel
```

6. 建立隧道：

```
ipsec auto --up mytunnel
```

7. 要在 **ipsec** 服务启动时自动启动隧道，请在连接定义中添加以下行：

```
auto=start
```

8. 如果您在带有 DHCP 或无状态地址自动配置(SLAAC)的网络中使用此主机，则连接可能会受到重定向的影响。有关详情和缓解步骤，请参阅[将 VPN 连接分配给专用路由表，以防止连接绕过隧道。](#)

## 7.5. 配置站点到站点的 VPN

要创建站点到站点的 IPsec VPN，通过加入两个网络，在两个主机之间创建一个 IPsec 隧道。主机因此充当端点，它们配置为允许来自一个或多个子网的流量通过。因此您可以将主机视为到网络远程部分的网关。

站点到站点 VPN 的配置只能与主机到主机 VPN 不同，同时必须在配置文件中指定一个或多个网络或子网。

### 先决条件

- 已配置了[主机到主机的 VPN](#)。

### 流程

1. 将带有主机到主机 VPN 配置的文件复制到新文件中，例如：

```
cp /etc/ipsec.d/my_host-to-host.conf /etc/ipsec.d/my_site-to-site.conf
```

2. 在上一步创建的文件中添加子网配置，例如：

```
conn mysubnet
 also=mytunnel
 leftsubnet=192.0.1.0/24
 rightsubnet=192.0.2.0/24
 auto=start

conn mysubnet6
 also=mytunnel
 leftsubnet=2001:db8:0:1::/64
 rightsubnet=2001:db8:0:2::/64
 auto=start

the following part of the configuration file is the same for both host-to-host and site-to-site
connections:

conn mytunnel
 leftid=@west
 left=192.1.2.23
 lefrtrsasigkey=0sAQOrlo+hOafUZDICQmXFrje/oZm [...] W2n417C/4urYHQkCvulQ==
 rightid=@east
 right=192.1.2.45
 rightrsasigkey=0sAQO3fwC6nSSGgt64DWiYZzuHbc4 [...] D/v8t5YTQ==
 authby=rsasig
```

3. 如果您在带有 DHCP 或无状态地址自动配置(SLAAC)的网络中使用此主机，则连接可能会受到重定向的影响。有关详情和缓解步骤，请参阅[将 VPN 连接分配给专用路由表，以防止连接绕过隧道。](#)

## 7.6. 配置远程访问 VPN

公路勇士是指拥有移动客户端和动态分配的 IP 地址的旅行用户。移动客户端使用 X.509 证书进行身份验证。

以下示例显示了 **IKEv2** 的配置，并且避免使用 **IKEv1 XAUTH** 协议。

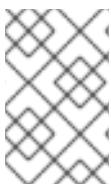
在服务器中：

```
conn roadwarriors
ikev2=insist
support (roaming) MOBIKE clients (RFC 4555)
mobike=yes
fragmentation=yes
left=1.2.3.4
if access to the LAN is given, enable this, otherwise use 0.0.0.0/0
leftsubnet=10.10.0.0/16
leftsubnet=0.0.0.0/0
leftcert=gw.example.com
leftid=%fromcert
leftxauthserver=yes
leftmodecfgserver=yes
right=%any
trust our own Certificate Agency
rightca=%same
pick an IP address pool to assign to remote users
100.64.0.0/16 prevents RFC1918 clashes when remote users are behind NAT
rightaddresspool=100.64.13.100-100.64.13.254
if you want remote clients to use some local DNS zones and servers
modecfgdns="1.2.3.4, 5.6.7.8"
modecfgdomains="internal.company.com, corp"
rightxauthclient=yes
rightmodecfgclient=yes
authby=rsasig
optionally, run the client X.509 ID through pam to allow or deny client
pam-authorize=yes
load connection, do not initiate
auto=add
kill vanished roadwarriors
dpddelay=1m
dpdtimeout=5m
dpdaction=clear
```

在移动客户端（即 road warrior 的设备）上，使用与之前配置稍有不同的配置：

```
conn to-vpn-server
ikev2=insist
pick up our dynamic IP
left=%defaultroute
leftsubnet=0.0.0.0/0
leftcert=myname.example.com
leftid=%fromcert
leftmodecfgclient=yes
right can also be a DNS hostname
right=1.2.3.4
```

```
if access to the remote LAN is required, enable this, otherwise use 0.0.0.0/0
rightsubnet=10.10.0.0/16
rightsubnet=0.0.0.0/0
fragmentation=yes
trust our own Certificate Agency
rightca=%same
authby=rsasig
allow narrowing to the server's suggested assigned IP and remote subnet
narrowing=yes
support (roaming) MOBIKE clients (RFC 4555)
mobike=yes
initiate connection
auto=start
```



### 注意

如果您在带有 DHCP 或无状态地址自动配置(SLAAC)的网络中使用此主机，则连接可能会受到重定向的影响。有关详情和缓解步骤，[请参阅将 VPN 连接分配给专用路由表，以防止连接绕过隧道。](#)

## 7.7. 配置网格 VPN

网格 VPN 网络（也称为 *any-to-any* VPN）是一个所有节点都使用 IPsec 进行通信的网络。该配置可以对于无法使用 IPsec 的节点进行例外处理。可使用两种方式配置网格 VPN 网络：

- 需要 IPsec。
- 首选 IPsec，但允许回退到使用明文通信。

节点之间的身份验证可以基于 X.509 证书或 DNS 安全扩展(DNSSEC)。

您对 *opportunistic IPsec* 使用任何常规的 IKEv2 验证方法，因为这些连接是常规的 Libreswan 配置，除了由 **right=%opportunisticgroup** 条目定义的 opportunistic IPsec 之外。常见的身份验证方法是主机使用常用共享认证机构(CA)根据 X.509 证书进行互相验证。作为标准流程的一部分，云部署通常为云中的每个节点发布证书。



### 重要

不要使用 PreSharedKey (PSK) 身份验证，因为一个有危险的主机会导致组 PSK secret 也有危险。

您可以使用 NULL 身份验证在节点间部署加密，而无需认证，这将防止被动攻击者。

以下流程使用 X.509 证书。您可以使用任何类型的 CA 管理系统（如 Dogtag 证书系统）生成这些证书。Dogtag 假设每个节点的证书都以 PKCS #12 格式(.p12 文件)提供，其包含私钥、节点证书和用于验证其他节点的 X.509 证书的根 CA 证书。

每个节点的配置与其 X.509 证书不同。这允许在不重新配置网络中的任何现有节点的情况下添加新节点。PKCS #12 文件需要一个“友好名称”，为此，我们使用名称“节点”，这样引用友好名称的配置文件对所有节点都是相同的。

### 先决条件

- Libreswan 已安装，并在每个节点上启动了 **ipsec** 服务。

- 一个新的 NSS 数据库已初始化。

- 如果您已经有一个旧的 NSS 数据库，请删除旧的数据库文件：

```
systemctl stop ipsec
rm /etc/ipsec.d/*db
```

- 您可以使用以下命令初始化新数据库：

```
ipsec initnss
```

## 步骤

- 在每个节点中导入 PKCS #12 文件。此步骤需要用于生成 PKCS #12 文件的密码：

```
ipsec import nodeXXX.p12
```

- 为 **IPsec 需要的**（专用）、**IPsec 可选的** (private-or-clear) 和 **No IPsec** (clear) 配置文件创建以下三个连接定义：

```
cat /etc/ipsec.d/mesh.conf
conn clear
auto=ondemand ①
type=passthrough
authby=never
left=%defaultroute
right=%group

conn private
auto=ondemand
type=transport
authby=rsasig
failureshunt=drop
negotiationshunt=drop
ikev2=insist
left=%defaultroute
leftcert=nodeXXXX
leftid=%fromcert ②
rightid=%fromcert
right=%opportunisticgroup

conn private-or-clear
auto=ondemand
type=transport
authby=rsasig
failureshunt=passthrough
negotiationshunt=passthrough
left
left=%defaultroute
leftcert=nodeXXXX ③
leftid=%fromcert
leftrsasigkey=%cert
right
```

```
rightrsaSigkey=%cert
rightid=%fromcert
right=%opportunisticgroup
```

### 1 auto 变量有几个选项：

您可以使用带有 opportunistic IPsec 的 **ondemand** 连接选项来启动 IPsec 连接，或者用于显式配置不需要一直激活的连接。这个选项在内核中建立一个陷阱 XFRM 策略，使 IPsec 连接在收到与该策略匹配的第一个数据包时开始。

您可以使用以下选项有效地配置和管理 IPsec 连接，无论是使用 Opportunistic IPsec 还是明确配置的连接：

#### add 选项

加载连接配置，并为响应远程启动做好准备。但是，连接不会自动从本地端启动。您可以使用 **ipsec auto --up** 命令手动启动 IPsec 连接。

#### start 选项

加载连接配置，并为响应远程启动做好准备。此外，它会立即启动到远程对等点的连接。您可以将这个选项用于永久的和一直活跃的连接。

### 2 leftid 和 rightid 变量标识 IPsec 隧道连接的右和左通道。如果您配置了证书，您可以使用这些变量来获取本地 IP 地址或本地证书的主题 DN 的值。

### 3 leftcert 变量定义您要使用的 NSS 数据库的昵称。

3. 将网络的 IP 地址添加到对应的类中。例如，如果所有节点都位于 **10.15.0.0/16** 网络中，则所有节点都必须使用 IPsec 加密：

```
echo "10.15.0.0/16" >> /etc/ipsec.d/policies/private
```

4. 要允许某些节点（如 **10.15.34.0/24**）使用或不使用 IPsec，请将这些节点添加到 private-or-clear 组中：

```
echo "10.15.34.0/24" >> /etc/ipsec.d/policies/private-or-clear
```

5. 要将一个不具有 IPsec 的主机（如 **10.15.1.2**）定义到 clear 组，请使用：

```
echo "10.15.1.2/32" >> /etc/ipsec.d/policies/clear
```

您可以通过模板为每个新节点在 **/etc/ipsec.d/policies** 目录中创建文件，或者使用 Puppet 或 Ansible 来提供它们。

请注意，每个节点都有相同的异常列表或不同的流量预期。因此，两个节点可能无法通信，因为一个节点需要 IPsec，而另一个节点无法使用 IPsec。

6. 重启节点将其添加到配置的网格中：

```
systemctl restart ipsec
```

7. 如果您在带有 DHCP 或无状态地址自动配置(SLAAC)的网络中使用此主机，则连接可能会受到重定向的影响。有关详情和缓解步骤，请参阅[将 VPN 连接分配给专用路由表，以防止连接绕过隧道](#)。

## 验证

- 使用 **ping** 命令打开 IPsec 隧道：

```
ping <nodeYYY>
```

- 显示带有导入认证的 NSS 数据库：

```
certutil -L -d sql:/etc/ipsec.d

Certificate Nickname Trust Attributes
SSL,S/MIME,JAR/XPI

west u,u,u
ca CT,,
```

- 查看节点上打开了哪个隧道：

```
ipsec trafficstatus
006 #2: "private#10.15.0.0/16"[1] ...<nodeYYY>, type=ESP, add_time=1691399301,
inBytes=512, outBytes=512, maxBytes=2^63B, id='C=US, ST=NC, O=Example
Organization, CN=east'
```

## 其他资源

- 您系统上的 **IPsec.conf (5)** 手册页
- 有关 **authby** 变量的更多信息，请参阅 [Libreswan 中的身份验证方法](#)。

## 7.8. 部署符合 FIPS 的 IPSEC VPN

您可以使用 Libreswan 部署符合 FIPS 的 IPsec VPN 解决方案。为此，您可以识别哪些加密算法可用，且在 FIPS 模式下，哪些算法对 Libreswan 禁用了。

### 先决条件

- AppStream** 存储库已启用。

### 流程

- 安装 **libreswan** 软件包：

```
yum install libreswan
```

- 如果您要重新安装 Libreswan，请删除其旧的 NSS 数据库：

```
systemctl stop ipsec
rm /etc/ipsec.d/*db
```

- 启动 **ipsec** 服务，并启用该服务，以便其在引导时自动启动：

```
systemctl enable ipsec --now
```

4. 通过添加 **ipsec** 服务，将防火墙配置为允许 IKE、ESP 的 **500** 和 **4500** UDP 端口，以及 AH 协议：

```
firewall-cmd --add-service="ipsec"
firewall-cmd --runtime-to-permanent
```

5. 将系统切换到 FIPS 模式：

```
fips-mode-setup --enable
```

6. 重启您的系统以允许内核切换到 FIPS 模式：

```
reboot
```

## 验证

1. 确认 Libreswan 在 FIPS 模式下运行：

```
ipsec whack --fipsstatus
000 FIPS mode enabled
```

2. 或者，检查 **systemd** 日志中的 **ipsec** 单元条目：

```
$ journalctl -u ipsec
...
Jan 22 11:26:50 localhost.localdomain pluto[3076]: FIPS Product: YES
Jan 22 11:26:50 localhost.localdomain pluto[3076]: FIPS Kernel: YES
Jan 22 11:26:50 localhost.localdomain pluto[3076]: FIPS Mode: YES
```

3. 以 FIPS 模式查看可用算法：

```
ipsec pluto --selftest 2>&1 | head -11
FIPS Product: YES
FIPS Kernel: YES
FIPS Mode: YES
NSS DB directory: sql:/etc/ipsec.d
Initializing NSS
Opening NSS database "sql:/etc/ipsec.d" read-only
NSS initialized
NSS crypto library initialized
FIPS HMAC integrity support [enabled]
FIPS mode enabled for pluto daemon
NSS library is running in FIPS mode
FIPS HMAC integrity verification self-test passed
```

4. 使用 FIPS 模式查询禁用的算法：

```
ipsec pluto --selftest 2>&1 | grep disabled
Encryption algorithm CAMELLIA_CTR disabled; not FIPS compliant
Encryption algorithm CAMELLIA_CBC disabled; not FIPS compliant
Encryption algorithm SERPENT_CBC disabled; not FIPS compliant
Encryption algorithm TWOFISH_CBC disabled; not FIPS compliant
Encryption algorithm TWOFISH_SSH disabled; not FIPS compliant
```

```

Encryption algorithm NULL disabled; not FIPS compliant
Encryption algorithm CHACHA20_POLY1305 disabled; not FIPS compliant
Hash algorithm MD5 disabled; not FIPS compliant
PRF algorithm HMAC_MD5 disabled; not FIPS compliant
PRF algorithm AES_XCBC disabled; not FIPS compliant
Integrity algorithm HMAC_MD5_96 disabled; not FIPS compliant
Integrity algorithm HMAC_SHA2_256_TRUNCBUG disabled; not FIPS compliant
Integrity algorithm AES_XCBC_96 disabled; not FIPS compliant
DH algorithm MODP1024 disabled; not FIPS compliant
DH algorithm MODP1536 disabled; not FIPS compliant
DH algorithm DH31 disabled; not FIPS compliant

```

5. 在 FIPS 模式中列出所有允许的算法和密码：

```

ipsec pluto --selftest 2>&1 | grep ESP | grep FIPS | sed "s/^.*FIPS//"
{256,192,*128} aes_ccm, aes_ccm_c
{256,192,*128} aes_ccm_b
{256,192,*128} aes_ccm_a
[*192] 3des
{256,192,*128} aes_gcm, aes_gcm_c
{256,192,*128} aes_gcm_b
{256,192,*128} aes_gcm_a
{256,192,*128} aesctr
{256,192,*128} aes
{256,192,*128} aes_gmac
sha, sha1, sha1_96, hmac_sha1
sha512, sha2_512, sha2_512_256, hmac_sha2_512
sha384, sha2_384, sha2_384_192, hmac_sha2_384
sha2, sha256, sha2_256, sha2_256_128, hmac_sha2_256
aes_cmac
null
null, dh0
dh14
dh15
dh16
dh17
dh18
ecp_256, ecp256
ecp_384, ecp384
ecp_521, ecp521

```

## 其他资源

- [使用系统范围的加密策略](#)

## 7.9. 使用密码保护 IPSEC NSS 数据库

默认情况下，IPsec 服务在第一次启动时使用空密码创建其网络安全服务(NSS)数据库。要提高安全性，您可以添加密码保护。



## 注意

在 RHEL 6.6 和之前的版本中，您必须使用一个密码来保护 IPsec NSS 数据库，以满足 FIPS 140-2 标准的要求，因为 NSS 加密库是针对 FIPS 140-2 Level 2 标准认证的。在 RHEL 8 中，NIS 将 NSS 认证为该标准级别 1，并且该状态不需要对数据库进行密码保护。

## 先决条件

- /etc/ipsec.d/ 目录包含 NSS 数据库文件。

## 流程

1. 为 Libreswan 的 **NSS** 数据库启用密码保护：

```
certutil -N -d sql:/etc/ipsec.d
Enter Password or Pin for "NSS Certificate DB":
Enter a password which will be used to encrypt your keys.
The password should be at least 8 characters long,
and should contain at least one non-alphabetic character.

Enter new password:
```

2. 创建包含您在上一步中设置的密码的 /etc/ipsec.d/nsspassword 文件，例如：

```
cat /etc/ipsec.d/nsspassword
NSS Certificate DB:_<password>_
```

**nsspassword** 文件使用以下语法：

```
<token_1>:<password1>
<token_2>:<password2>
```

默认的 NSS 软件令牌是 **NSS 证书数据库**。如果您的系统以 FIPS 模式运行，则令牌的名称为 **NSS FIPS 140-2 证书数据库**。

3. 根据您的场景，在完成了 **nsspassword** 文件后，启动或重启 **ipsec** 服务：

```
systemctl restart ipsec
```

## 验证

1. 在其 NSS 数据库中添加非空密码后，检查 **ipsec** 服务是否运行：

```
systemctl status ipsec
● ipsec.service - Internet Key Exchange (IKE) Protocol Daemon for IPsec
 Loaded: loaded (/usr/lib/systemd/system/ipsec.service; enabled; vendor preset: disable)
 Active: active (running)...
```

2. 检查 **Journal** 日志是否包含确认成功初始化的条目：

```
journalctl -u ipsec
...
pluto[6214]: Initializing NSS using read-write database "sql:/etc/ipsec.d"
```

```
pluto[6214]: NSS Password from file "/etc/ipsec.d/nsspassword" for token "NSS Certificate DB" with length 20 passed to NSS
pluto[6214]: NSS crypto library initialized
...

```

## 其他资源

- 您系统上的 **certutil (1)** 手册页
- [合规性活动和政府标准](#) 知识库文章中的 FIPS 140-2 和 FIPS 140-3。

## 7.10. 配置 IPSEC VPN 以使用 TCP

Libreswan 支持 IKE 和 IPsec 数据包的 TCP 封装，如 RFC 8229 所述。有了这个功能，您可以在网络上建立 IPsec VPN，以防止通过 UDP 和封装安全负载(ESP)传输的流量。您可以将 VPN 服务器和客户端配置为使用 TCP 作为回退，或者作为主 VPN 传输协议。由于 TCP 封装的性能成本较高，因此只有在您的场景中需要永久阻止 UDP 时，才使用 TCP 作为主 VPN 协议。

### 先决条件

- 已配置了 [远程访问 VPN](#)。

### 流程

1. 在 `/etc/ipsec.conf` 文件的 **config setup** 部分中添加以下选项：

```
listen-tcp=yes
```

2. 要在第一次尝试 UDP 失败时使用 TCP 封装作为回退选项，请在客户端的连接定义中添加以下两个选项：

```
enable-tcp=fallback
tcp-remoteport=4500
```

另外，如果您知道 UDP 会被永久阻止，请在客户端的连接配置中使用以下选项：

```
enable-tcp=yes
tcp-remoteport=4500
```

### 其他资源

- [IETF RFC 8229：IKE 和 IPsec 数据包的 TCP 封装](#)

## 7.11. 配置自动检测和使用 ESP 硬件卸载来加速 IPSEC 连接

卸载硬件的封装安全负载(ESP)来加速以太网上的 IPsec 连接。默认情况下，Libreswan 会检测硬件是否支持这个功能，并因此启用 ESP 硬件卸载。如果这个功能被禁用或明确启用，您可以切回到自动检测。

### 先决条件

- 网卡支持 ESP 硬件卸载。

- 网络驱动程序支持 ESP 硬件卸载。
- IPsec 连接已配置且可以正常工作。

## 步骤

1. 编辑连接的 `/etc/ipsec.d/` 目录中的 Libreswan 配置文件，该文件应使用 ESP 硬件卸载支持的自动检测。
2. 确保连接的设置中没有设置 **nic-offload** 参数。
3. 如果您删除了 **nic-offload**，请重启 **ipsec** 服务：

```
systemctl restart ipsec
```

## 验证

1. 显示 IPsec 连接使用的以太网设备的 **tx\_ipsec** 和 **rx\_ipsec** 计数器：

```
ethtool -S enp1s0 | grep -E "_ipsec"
tx_ipsec: 10
rx_ipsec: 10
```

2. 通过 IPsec 隧道发送流量。例如，ping 远程 IP 地址：

```
ping -c 5 remote_ip_address
```

3. 再次显示以太网设备的 **tx\_ipsec** 和 **rx\_ipsec** 计数器：

```
ethtool -S enp1s0 | grep -E "_ipsec"
tx_ipsec: 15
rx_ipsec: 15
```

如果计数器值增加了，ESP 硬件卸载正常工作。

## 其他资源

- [使用 IPsec 配置 VPN](#)

## 7.12. 在绑定中配置 ESP 硬件卸载以加快 IPSEC 连接

将封装安全负载(ESP)卸载到硬件可加速 IPsec 连接。如果出于故障转移原因而使用网络绑定，配置 ESP 硬件卸载的要求和流程与使用常规以太网设备的要求和流程不同。例如，在这种情况下，您可以对绑定启用卸载支持，内核会将设置应用到绑定的端口。

### 先决条件

- 绑定中的所有网卡都支持 ESP 硬件卸载。使用 `ethtool -k <interface_name> | grep "esp-hw-offload"` 命令验证每个绑定端口是否都支持此功能。
- 绑定已配置且可以正常工作。
- 该绑定使用 **active-backup** 模式。绑定驱动程序不支持此功能的任何其他模式。

- IPsec 连接已配置且可以正常工作。

## 步骤

- 对网络绑定启用 ESP 硬件卸载支持：

```
nmcli connection modify bond0 ethtool.feature-esp-hw-offload on
```

这个命令在对 **bond0** 连接启用 ESP 硬件卸载支持。

- 重新激活 **bond0** 连接：

```
nmcli connection up bond0
```

- 编辑应使用 ESP 硬件卸载的连接的 **/etc/ipsec.d/** 目录中的 Libreswan 配置文件，并将 **nic-offload=yes** 语句附加到连接条目：

```
conn example
...
nic-offload=yes
```

- 重启 **ipsec** 服务：

```
systemctl restart ipsec
```

## 验证

验证方法取决于各个方面，如内核版本和驱动程序。例如，某些驱动程序提供计数器，但它们的名称可能会有所不同。详情请查看网络驱动程序文档。

以下验证步骤适用于 Red Hat Enterprise Linux 8 上的 **ixgbe** 驱动程序：

- 显示绑定的活动端口：

```
grep "Currently Active Slave" /proc/net/bonding/bond0
Currently Active Slave: enp1s0
```

- 显示活动端口的 **tx\_ipsec** 和 **rx\_ipsec** 计数器：

```
ethtool -S enp1s0 | grep -E "_ipsec"
tx_ipsec: 10
rx_ipsec: 10
```

- 通过 IPsec 隧道发送流量。例如，ping 远程 IP 地址：

```
ping -c 5 remote_ip_address
```

- 再次显示活动端口的 **tx\_ipsec** 和 **rx\_ipsec** 计数器：

```
ethtool -S enp1s0 | grep -E "_ipsec"
tx_ipsec: 15
rx_ipsec: 15
```

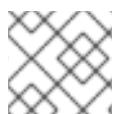
如果计数器值增加了，ESP 硬件卸载正常工作。

## 其他资源

- 配置网络绑定
- 设置 IPsec VPN

### 7.13. 使用 RHEL 系统角色配置 IPSEC VPN 连接

VPN 是一个加密连接，用于通过不受信任的网络安全地传输流量。通过使用 **vpn** RHEL 系统角色，您可以自动执行创建 VPN 配置的过程。



#### 注意

**vpn** RHEL 系统角色只支持 Libreswan（一个 IPsec 实现），作为 VPN 供应商。

#### 7.13.1. 使用 **vpn** RHEL 系统角色使用 PSK 身份验证配置 IPsec 主机到主机的 VPN

您可以使用 IPsec 通过 VPN 直接连接主机。主机可以使用预共享密钥(PSK)来相互进行身份验证。通过使用 **vpn** RHEL 系统角色，您可以使用 PSK 身份验证自动创建 IPsec 主机到主机连接的过程。

默认情况下，该角色创建基于隧道的 VPN。

#### 先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户对其具有 **sudo** 权限。

#### 步骤

1. 创建一个包含以下内容的 playbook 文件，如 **~/playbook.yml**：

```

- name: Configuring VPN
 hosts: managed-node-01.example.com, managed-node-02.example.com
 tasks:
 - name: IPsec VPN with PSK authentication
 ansible.builtin.include_role:
 name: redhat.rhel_system_roles.vpn
 vars:
 vpn_connections:
 - hosts:
 managed-node-01.example.com:
 managed-node-02.example.com:
 auth_method: psk
 auto: start
 vpn_manage_firewall: true
 vpn_manage_selinux: true
```

示例 playbook 中指定的设置包括以下内容：

**hosts: <list>**

使用您要配置 VPN 的主机定义 YAML 字典。如果条目不是 Ansible 受管节点，您必须在 **hostname** 参数中指定其完全限定域名(FQDN)或 IP 地址，例如：

```
...
- hosts:
 ...
 external-host.example.com:
 hostname: 192.0.2.1
```

该角色在每个受管节点上配置 VPN 连接。连接名为 <**host\_A**>-to-<**host\_B**>，例如 **managed-node-01.example.com-to-managed-node-02.example.com**。请注意，该角色不能在外部（未管理）节点上配置 Libreswan。您必须在这些主机上手动创建配置。

**auth\_method: psk**

启用主机间的 PSK 身份验证。该角色使用控制节点上的 **openssl** 创建 PSK。

**auto: <start-up\_method>**

指定连接的启动方法。有效值包括 **添加**、**ondemand**、**start** 和 **ignore**。详情请查看安装了 Libreswan 的系统上的 **ipsec.conf (5)** 手册页。此变量的默认值为 null，这意味着没有自动启动操作。

**vpn\_manage\_firewall: true**

定义该角色在受管节点上的 **firewalld** 服务中打开所需的端口。

**vpn\_manage\_selinux: true**

定义该角色在 IPsec 端口上设置所需的 SELinux 端口类型。

有关 playbook 中使用的所有变量的详情，请查看控制节点上的 **/usr/share/ansible/roles/rhel-system-roles.vpn/README.md** 文件。

2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

## 验证

- 确认连接成功启动，例如：

```
ansible managed-node-01.example.com -m shell -a 'ipsec trafficstatus | grep "managed-node-01.example.com-to-managed-node-02.example.com"'
...
006 #3: "managed-node-01.example.com-to-managed-node-02.example.com", type=ESP,
add_time=1741857153, inBytes=38622, outBytes=324626, maxBytes=2^63B,
id='@managed-node-02.example.com'
```

请注意，只有在 VPN 连接活跃时，这个命令才会成功。如果将 playbook 中的 **auto** 变量设置为 **start** 以外的值，您可能需要首先手动激活受管节点上的连接。

## 其他资源

- [/usr/share/ansible/roles/rhel-system-roles.vpn/README.md](#) 文件
- [/usr/share/doc/rhel-system-roles/vpn/](#) 目录

### 7.13.2. 使用 PSK 身份验证配置 IPsec 主机到主机的 VPN，并使用 `vpn` RHEL 系统角色分隔数据和 control plane

您可以使用 IPsec，通过 VPN 直接将主机互联。例如，要通过最大程度减少被截获或破坏的控制消息的风险来增强安全性，您可以为数据流量和控制流量配置单独的连接。通过使用 `vpn` RHEL 系统角色，您可以自动化创建带有单独的数据和控制平面及 PSK 身份验证的 IPsec 主机到主机连接的过程。

#### 先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 您用于连接到受管节点的帐户对它们具有 `sudo` 权限。

#### 流程

1. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```

- name: Configuring VPN
 hosts: managed-node-01.example.com, managed-node-02.example.com
 tasks:
 - name: IPsec VPN with PSK authentication
 ansible.builtin.include_role:
 name: redhat.rhel_system_roles.vpn
 vars:
 vpn_connections:
 - name: control_plane_vpn
 hosts:
 managed-node-01.example.com:
 hostname: 203.0.113.1 # IP address for the control plane
 managed-node-02.example.com:
 hostname: 198.51.100.2 # IP address for the control plane
 auth_method: psk
 auto: start
 - name: data_plane_vpn
 hosts:
 managed-node-01.example.com:
 hostname: 10.0.0.1 # IP address for the data plane
 managed-node-02.example.com:
 hostname: 172.16.0.2 # IP address for the data plane
 auth_method: psk
 auto: start
 vpn_manage_firewall: true
 vpn_manage_selinux: true
```

示例 playbook 中指定的设置包括如下：

**hosts: <list>**

定义一个 YAML 字典，其中包含您想要在它们之间配置 VPN 的主机。连接被命名为 **<name>-<IP\_address\_A>-to-<IP\_address\_B>**，例如 **control\_plane\_vpn-203.0.113.1-to-198.51.100.2**。

角色在每个受管节点上配置 VPN 连接。请注意，角色不能在外部（未管理的）节点上配置 Libreswan。您必须在这些主机上手动创建配置。

**auth\_method: psk**

在主机之间启用 PSK 身份验证。角色在控制节点上使用 **openssl** 来创建预共享密钥。

**auto: <start-up\_method>**

指定连接的启动方法。有效值是 **add**、**ondemand**、**start** 和 **ignore**。详情请查看安装了 Libreswan 的系统上的 **ipsec.conf (5)** 手册页。此变量的默认值为 **null**，这意味着没有自动启动操作。

**vpn\_manage\_firewall: true**

定义角色在受管节点上的 **firewalld** 服务中打开所需的端口。

**vpn\_manage\_selinux: true**

定义角色在 IPsec 端口上设置所需的 SELinux 端口类型。

有关 playbook 中使用的所有变量的详情，请查看控制节点上的 **/usr/share/ansible/roles/rhel-system-roles.vpn/README.md** 文件。

## 2. 验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不能防止错误的、但有效的配置。

## 3. 运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

## 验证

- 确认连接已成功启动，例如：

```
ansible managed-node-01.example.com -m shell -a 'ipsec trafficstatus | grep "control_plane_vpn-203.0.113.1-to-198.51.100.2"'
...
006 #3: "control_plane_vpn-203.0.113.1-to-198.51.100.2", type=ESP,
add_time=1741860073, inBytes=0, outBytes=0, maxBytes=2^63B, id='198.51.100.2'
```

请注意，只有在 VPN 连接活跃时，这个命令才会成功。如果将 playbook 中的 **auto** 变量设置为 **start** 以外的值，您可能需要首先手动激活受管节点上的连接。

## 其他资源

- /usr/share/ansible/roles/rhel-system-roles.vpn/README.md** 文件
- /usr/share/doc/rhel-system-roles/vpn/** 目录

### 7.13.3. 使用 vpn RHEL 系统角色使用基于证书的身份验证配置 IPsec 网格 VPN

Libreswan 支持创建一个机会网格，以便在每个主机上带有单个配置的大量主机之间建立 IPsec 连接。将主机添加到网格中不需要更新现有主机上的配置。为提高安全性，请在 Libreswan 中使用基于证书的身份验证。

通过使用 **vpn** RHEL 系统角色，您可以在受管节点之间自动化配置带有基于证书的身份验证的 VPN 网格。

#### 先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 您用于连接到受管节点的帐户对它们具有 **sudo** 权限。
- 您可以为每个受管节点准备一个 PKCS #12 文件：
  - 每个文件包含：
    - 证书颁发机构(CA)证书
    - 节点的私钥
    - 节点的客户端证书
  - 文件被命名为 **<managed\_node\_name\_as\_in\_the\_inventory>.p12**。
  - 文件存储在与 playbook 相同的目录中。

#### 流程

1. 编辑 **~/inventory** 文件，并附加 **cert\_name** 变量：

```
managed-node-01.example.com cert_name=managed-node-01.example.com
managed-node-02.example.com cert_name=managed-node-02.example.com
managed-node-03.example.com cert_name=managed-node-03.example.com
```

将 **cert\_name** 变量设置为每个主机的证书中使用的通用名称(CN)字段的值。通常，CN 字段被设置为完全限定域名(FQDN)。

2. 将敏感变量存储在加密的文件中：

- a. 创建 vault：

```
$ ansible-vault create ~/vault.yml
New Vault password: <vault_password>
Confirm New Vault password: <vault_password>
```

- b. 在 **ansible-vault create** 命令打开编辑器后，以 **<key>: <value>** 格式输入敏感数据：

```
pkcs12_pwd: <password>
```

- c. 保存更改，并关闭编辑器。Ansible 加密 vault 中的数据。

3. 创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```
- name: Configuring VPN
 hosts: managed-node-01.example.com, managed-node-02.example.com, managed-node-03.example.com
 vars_files:
 - ~/vault.yml
 tasks:
 - name: Install LibreSwan
 ansible.builtin.package:
 name: libreswan
 state: present

 - name: Identify the path to IPsec NSS database
 ansible.builtin.set_fact:
 nss_db_dir: "{{ '/etc/ipsec.d/' if
 ansible_distribution in ['CentOS', 'RedHat']
 and ansible_distribution_major_version is version('8', '=')
 else '/var/lib/ipsec/nss/' }}"

 - name: Locate IPsec NSS database files
 ansible.builtin.find:
 paths: "{{ nss_db_dir }}"
 patterns: "*.db"
 register: db_files

 - name: Remove IPsec NSS database files
 ansible.builtin.file:
 path: "{{ item.path }}"
 state: absent
 loop: "{{ db_files.files }}"
 when: db_files.matched > 0

 - name: Initialize IPsec NSS database
 ansible.builtin.command:
 cmd: ipsec initnss

 - name: Copy PKCS #12 file to the managed node
 ansible.builtin.copy:
 src: "~/{{ inventory_hostname }}.p12"
 dest: "/etc/ipsec.d/{{ inventory_hostname }}.p12"
 mode: 0600

 - name: Import PKCS #12 file in IPsec NSS database
 ansible.builtin.shell:
 cmd: 'pk12util -d {{ nss_db_dir }} -i /etc/ipsec.d/{{ inventory_hostname }}.p12 -W "{{ pkcs12_pwd }}"'

 - name: Remove PKCS #12 file
 ansible.builtin.file:
 path: "/etc/ipsec.d/{{ inventory_hostname }}.p12"
 state: absent

 - name: Opportunistic mesh IPsec VPN with certificate-based authentication
 ansible.builtin.include_role:
 name: redhat.rhel_system_roles.vpn
```

```

vars:
 vpn_connections:
 - opportunistic: true
 auth_method: cert
 policies:
 - policy: private
 cidr: default
 - policy: private
 cidr: 192.0.2.0/24
 - policy: clear
 cidr: 192.0.2.1/32
 vpn_manage_firewall: true
 vpn_manage_selinux: true

```

示例 playbook 中指定的设置包括如下：

#### **opportunistic: true**

在多个主机之间启用机会网格。 **policies** 变量定义哪些子网和主机流量必须或可以加密，它们中哪些应该继续使用明文连接。

#### **auth\_method: cert**

启用基于证书的身份验证。这要求您在清单中指定每个受管节点的证书的昵称。

#### **policies: <list\_of\_policies>**

以 YAML 列表格式定义 Libreswan 策略。

默认策略是 **private-or-clear**。要将它改为 **private**，上面的 playbook 包含默认 **cidr** 条目的一个相应策略。

如果 Ansible 控制节点与受管节点在同一个 IP 子网中，要防止执行 playbook 期间丢失 SSH 连接，请为控制节点的 IP 地址添加一个 **clear** 策略。例如，如果应该为 **192.0.2.0/24** 子网配置网格，并且控制节点使用 IP 地址 **192.0.2.1**，则您需要一个用于**192.0.2.1/32** 的 **clear** 策略，如 playbook 中所示。

有关策略的详情，请查看安装了 Libreswan 的系统上的 **ipsec.conf (5)** 手册页。

#### **vpn\_manage\_firewall: true**

定义角色在受管节点上的 **firewalld** 服务中打开所需的端口。

#### **vpn\_manage\_selinux: true**

定义角色在 IPsec 端口上设置所需的 SELinux 端口类型。

有关 playbook 中使用的所有变量的详情，请查看控制节点上的 **/usr/share/ansible/roles/rhel-system-roles.vpn/README.md** 文件。

## 4. 验证 playbook 语法：

```
$ ansible-playbook --ask-vault-pass --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不能防止错误的、但有效的配置。

## 5. 运行 playbook：

```
$ ansible-playbook --ask-vault-pass ~/playbook.yml
```

- 在网格的一个节点上， ping 另一个节点来激活连接：

```
[root@managed-node-01]# ping managed-node-02.example.com
```

- 确认连接是否处于活跃状态：

```
[root@managed-node-01]# ipsec trafficstatus
006 #2: "private#192.0.2.0/24"[1] ...192.0.2.2, type=ESP, add_time=1741938929,
inBytes=372408, outBytes=545728, maxBytes=2^63B, id='CN=managed-node-02.example.com'
```

## 其他资源

- [/usr/share/ansible/roles/rhel-system-roles.vpn/README.md](#) 文件
- [/usr/share/doc/rhel-system-roles/vpn/](#) 目录

## 7.14. 配置选择不使用系统范围的加密策略的 IPSEC 连接

### 为连接覆盖系统范围的加密策略

RHEL 系统范围的加密策略会创建一个名为 **%default** 的特殊连接。此连接包含 **ikev2**、**esp** 和 **ike** 选项的默认值。但是，您可以通过在连接配置文件中指定上述选项来覆盖默认值。

例如，以下配置允许使用带有 AES 和 SHA-1 或 SHA-2 的 IKEv1 连接，以及带有 AES-GCM 或 AES-CBC 的 IPsec(ESP) 连接：

```
conn MyExample
...
ikev2=never
ike=aes-sha2,aes-sha1;modp2048
esp=aes_gcm,aes-sha2,aes-sha1
...
```

请注意，AES-GCM 可用于 IPsec(ESP) 和 IKEv2，但不适用于 IKEv1。

### 为所有连接禁用系统范围的加密策略

要禁用所有 IPsec 连接的系统范围的加密策略，请在 **/etc/ipsec.conf** 文件中注释掉以下行：

```
include /etc/crypto-policies/back-ends/libreswan.config
```

然后将 **ikev2=never** 选项添加到连接配置文件。

## 其他资源

- [使用系统范围的加密策略](#)

## 7.15. IPSEC VPN 配置故障排除

与 IPsec VPN 配置相关的问题通常是由于几个主要原因造成的。如果您遇到此类问题，您可以检查问题的原因是否符合一下任何一种情况，并应用相应的解决方案。

## 基本连接故障排除

VPN连接的大多数问题都发生在新部署中，管理员使用不匹配的配置选项配置了端点。此外，正常工作的配置可能会突然停止工作，通常是由于新引入的不兼容的值。这可能是管理员更改配置的结果。或者，管理员可能已安装了固件更新，或者使用某些选项的不同默认值（如加密算法）安装了软件包更新。

要确认已建立IPsec VPN连接：

```
ipsec trafficstatus
006 #8: "vpn.example.com"[1] 192.0.2.1, type=ESP, add_time=1595296930, inBytes=5999,
outBytes=3231, id='@vpn.example.com', lease=100.64.13.5/32
```

如果输出为空或者没有显示具有连接名称的条目，则隧道将断开。

检查连接中的问题：

- 重新载入 *vpn.example.com* 连接：

```
ipsec auto --add vpn.example.com
002 added connection description "vpn.example.com"
```

- 下一步，启动VPN连接：

```
ipsec auto --up vpn.example.com
```

## 与防火墙相关的问题

最常见的问题是，其中一个IPsec端点或端点之间路由器上的防火墙将所有互联网密钥交换(IKE)数据包丢弃。

- 对于IKEv2，类似以下示例的输出说明防火墙出现问题：

```
ipsec auto --up vpn.example.com
181 "vpn.example.com"[1] 192.0.2.2 #15: initiating IKEv2 IKE SA
181 "vpn.example.com"[1] 192.0.2.2 #15: STATE_PARENT_I1: sent v2I1, expected v2R1
010 "vpn.example.com"[1] 192.0.2.2 #15: STATE_PARENT_I1: retransmission; will wait 0.5
seconds for response
010 "vpn.example.com"[1] 192.0.2.2 #15: STATE_PARENT_I1: retransmission; will wait 1
seconds for response
010 "vpn.example.com"[1] 192.0.2.2 #15: STATE_PARENT_I1: retransmission; will wait 2
seconds for
...
...
```

- 对于IKEv1，启动命令的输出如下：

```
ipsec auto --up vpn.example.com
002 "vpn.example.com" #9: initiating Main Mode
102 "vpn.example.com" #9: STATE_MAIN_I1: sent MI1, expecting MR1
010 "vpn.example.com" #9: STATE_MAIN_I1: retransmission; will wait 0.5 seconds for
response
010 "vpn.example.com" #9: STATE_MAIN_I1: retransmission; will wait 1 seconds for
response
010 "vpn.example.com" #9: STATE_MAIN_I1: retransmission; will wait 2 seconds for
response
...
...
```

由于用于设置 IPsec 的 IKE 协议已经加密，因此您只能使用 **tcpdump** 工具排除一小部分问题。如果防火墙丢弃了 IKE 或 IPsec 数据包，您可以尝试使用 **tcpdump** 工具来查找原因。但是，**tcpdump** 无法诊断 IPsec VPN 连接的其他问题。

- 捕获**eth0** 接口上的 VPN 协商以及所有加密数据：

```
tcpdump -i eth0 -n -n esp or udp port 500 or udp port 4500 or tcp port 4500
```

## 不匹配的算法、协议和策略

VPN 连接要求端点具有匹配的 IKE 算法、IPsec 算法和 IP 地址范围。如果发生不匹配，连接会失败。如果您使用以下方法之一发现不匹配，请通过匹配算法、协议或策略来修复它。

- 如果远程端点没有运行 IKE/IPsec，您可以看到一个 ICMP 数据包来指示它。例如：

```
ipsec auto --up vpn.example.com
...
000 "vpn.example.com"[1] 192.0.2.2 #16: ERROR: asynchronous network error report on
wlp2s0 (192.0.2.2:500), complainant 198.51.100.1: Connection refused [errno 111, origin
ICMP type 3 code 3 (not authenticated)]
...
```

- 不匹配 IKE 算法示例：

```
ipsec auto --up vpn.example.com
...
003 "vpn.example.com"[1] 193.110.157.148 #3: dropping unexpected IKE_SA_INIT message
containing NO_PROPOSAL_CHOSEN notification; message payloads: N; missing payloads:
SA,KE,Ni
```

- 不匹配 IPsec 算法示例：

```
ipsec auto --up vpn.example.com
...
182 "vpn.example.com"[1] 193.110.157.148 #5: STATE_PARENT_I2: sent v2I2, expected
v2R2 {auth=IKEv2 cipher=AES_GCM_16_256 integ=n/a prf=HMAC_SHA2_256
group=MODP2048}
002 "vpn.example.com"[1] 193.110.157.148 #6: IKE_AUTH response contained the error
notification NO_PROPOSAL_CHOSEN
```

不匹配的 IKE 版本还可导致远程端点在没有响应的情况下丢弃请求。这与丢弃所有 IKE 数据包的防火墙相同。

- IKEv2 不匹配的 IP 地址范围示例（称为流量选择器 - TS）：

```
ipsec auto --up vpn.example.com
...
1v2 "vpn.example.com" #1: STATE_PARENT_I2: sent v2I2, expected v2R2 {auth=IKEv2
cipher=AES_GCM_16_256 integ=n/a prf=HMAC_SHA2_512 group=MODP2048}
002 "vpn.example.com" #2: IKE_AUTH response contained the error notification
TS_UNACCEPTABLE
```

- IKEv1 的不匹配 IP 地址范围示例：

```
ipsec auto --up vpn.example.com
...
031 "vpn.example.com" #2: STATE_QUICK_I1: 60 second timeout exceeded after 0
retransmits. No acceptable response to our first Quick Mode message: perhaps peer likes
no proposal
```

- 当在 IKEv1 中使用预共享密钥(PSK)时，如果双方没有放入相同的 PSK，则整个 IKE 信息将无法读取：

```
ipsec auto --up vpn.example.com
...
003 "vpn.example.com" #1: received Hash Payload does not match computed value
223 "vpn.example.com" #1: sending notification INVALID_HASH_INFORMATION to
192.0.2.23:500
```

- 在 IKEv2 中，不匹配-PSK 错误会导致 AUTHENTICATION\_FAILED 信息：

```
ipsec auto --up vpn.example.com
...
002 "vpn.example.com" #1: IKE SA authentication request rejected by peer:
AUTHENTICATION_FAILED
```

## 最大传输单元

除防火墙阻止 IKE 或 IPsec 数据包外，网络问题的最常见原因与加密数据包的数据包大小增加有关。网络硬件对于大于最大传输单元(MTU)的数据包进行分片处理，例如 1500 字节。通常，片会丢失，数据包无法重新组装。当使用小数据包的 ping 测试可以正常工作，但其他流量失败时，这会导致间歇性故障。在这种情况下，您可以建立一个 SSH 会话，但是一使用它，终端就会冻结，例如，在远程主机上输入 'ls -al /usr' 命令。

要临时解决这个问题，请通过将 **mtu=1400** 选项添加到隧道配置文件中来减小 MTU 大小。

另外，对于 TCP 连接，启用更改 MSS 值的 iptables 规则：

```
iptables -I FORWARD -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --clamp-mss-to-pmtu
```

如果上一命令没有解决您场景中的问题，请在 **set-mss** 参数中直接指定较小的数值：

```
iptables -I FORWARD -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --set-mss 1380
```

## 网络地址转换(NAT)

当 IPsec 主机也充当 NAT 路由器时，可能会意外地重新映射数据包。以下示例配置演示了这个问题：

```
conn myvpn
left=172.16.0.1
leftsubnet=10.0.2.0/24
right=172.16.0.2
rightsubnet=192.168.0.0/16
...
```

地址为 172.16.0.1 的系统有一个 NAT 规则：

```
iptables -t nat -I POSTROUTING -o eth0 -j MASQUERADE
```

如果地址为 10.0.2.33 的系统将数据包发送到 192.168.0.1，那么路由器会在应用 IPsec 加密前将源 10.0.2.33 转换为 172.16.0.1。

然后，源地址为 10.0.2.33 的数据包不再与 **conn myvpn** 配置匹配，IPsec 不会加密此数据包。

要解决这个问题，请在路由器上插入目标 IPsec 子网范围不包含 NAT 的规则，例如：

```
iptables -t nat -I POSTROUTING -s 10.0.2.0/24 -d 192.168.0.0/16 -j RETURN
```

## 内核 IPsec 子系统错误

例如，当 bug 导致 IKE 用户空间和 IPsec 内核不同步时，内核 IPsec 子系统可能会失败。检查此问题：

```
$ cat /proc/net/xfrm_stat
XfrmInError 0
XfrmInBufferError 0
...
...
```

上一命令输出中的任何非零值都表示有问题。如果您遇到这个问题，请开一个新的 [支持问题单](#)，并附上上一命令的输出与对应的 IKE 日志。

## libreswan 日志

默认情况下，Libreswan 使用 **syslog** 协议的日志。您可以使用 **journalctl** 命令来查找与 IPsec 有关的日志条目。因为日志中相应的条目是由 **pluto** IKE 守护进程发送的，因此请搜索 "pluto" 关键字，例如：

```
$ journalctl -b | grep pluto
```

显示 **ipsec** 服务的实时日志：

```
$ journalctl -f -u ipsec
```

如果默认日志记录级别没有显示您的配置问题，请将 **plutodebug=all** 选项添加到 **/etc/ipsec.conf** 文件的 **config setup** 部分来启用调试日志。

请注意，调试日志记录会生成大量的条目，**journalctl** 或 **syslogd** 服务的速率可能会抑制 **syslog** 消息。要确保您有完整的日志，请将日志记录重定向到文件中。编辑 **/etc/ipsec.conf**，并在 **config setup** 部分中添加 **logfile=/var/log/pluto.log**。

## 其他资源

- [使用日志文件对问题进行故障排除](#)
- [tcpdump\(8\) 和 ipsec.conf\(5\) 手册页。](#)
- [使用和配置 firewalld](#)

## 7.16. 使用 CONTROL-CENTER 配置 VPN 连接

如果使用带有图形界面的 Red Hat Enterprise Linux，您可以在 GNOME **control-center** 中配置 VPN 连接。

## 先决条件

- 已安装 **NetworkManager-libreswan-gnome** 软件包。

## 流程

1. 按 **Super** 键，输入 **Settings**，然后按 **Enter** 键打开 **control-center** 应用程序。
2. 选择左侧的 **Network** 条目。
3. 点 **+** 图标。
4. 选择 **VPN**。
5. 选择 **Identity** 菜单项来查看基本配置选项：  
**General**  
gateway- 远程 VPN 网关的名称或 **IP** 地址。

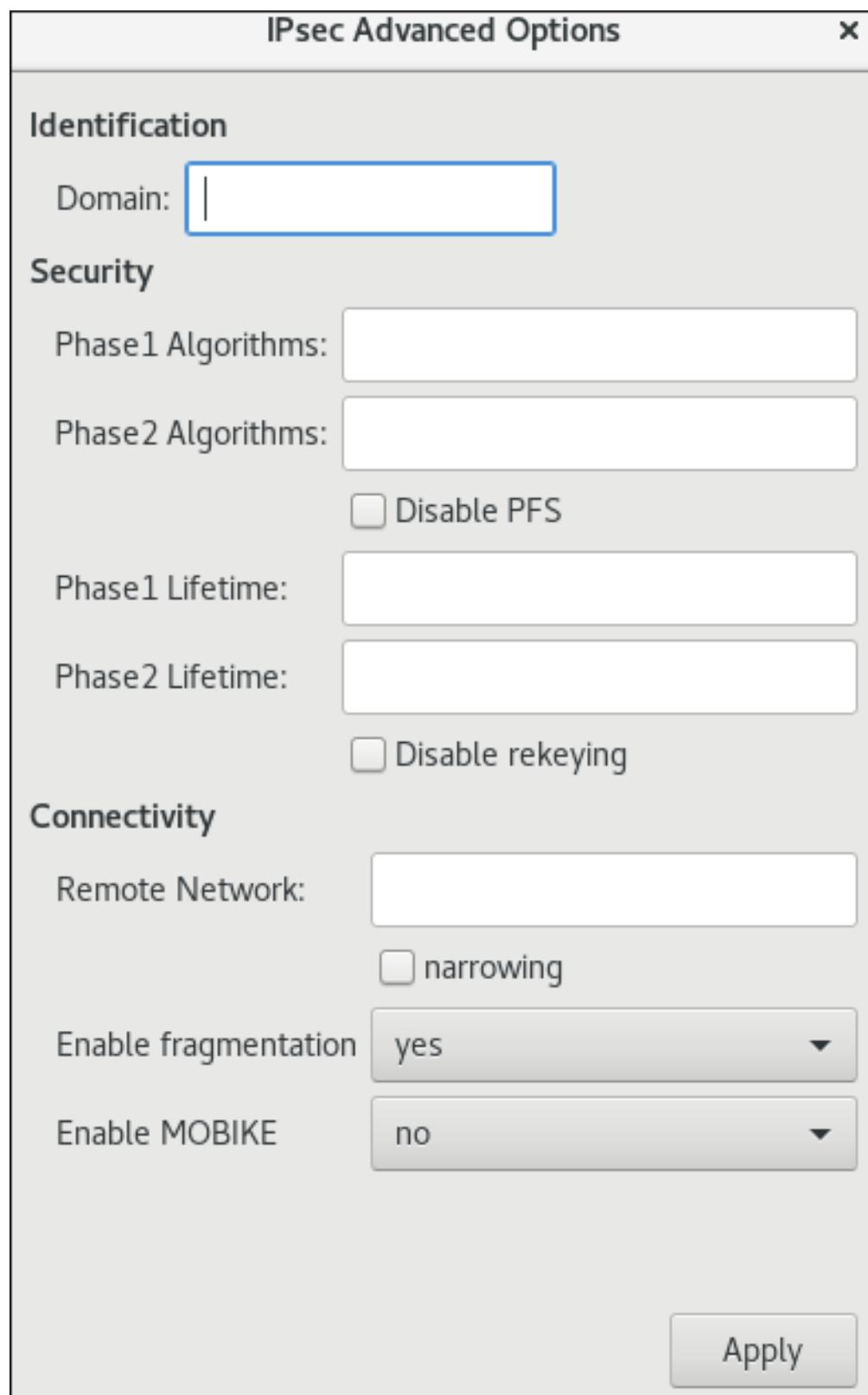
## 认证

## 类型

- **IKEv2(Certificate)**- 客户端通过证书进行身份验证。它更安全（默认）。
- **IKEv1(XAUTH)** - 客户端通过用户名和密码或预共享密钥(**PSK**)进行身份验证。

以下配置设置在 **高级** 部分中提供：

图 7.1. VPN 连接的高级选项





### 警告

当使用 `gnome-control-center` 应用程序配置基于 IPsec 的 VPN 连接时，高级对话框会显示配置，但它不允许任何更改。因此，用户无法更改任何高级 IPsec 选项。使用 `nm-connection-editor` 或 `nmcli` 工具来配置高级属性。

### 身份识别

- 域 -如果需要，请输入域名。

### 安全性

- **Phase1 Algorithms** - 对应于 `ike Libreswan` 参数 - 输入用来验证和设置加密频道的算法。
- **Phase2 Algorithms** - 对应于 `esp Libreswan` 参数 - 输入用于 IPsec 协商的算法。

选择 **Disable PFS** 字段来关闭 Perfect Forward Secrecy(PFS)，以确保与不支持 PFS 的旧服务器兼容。

- **Phase1 Lifetime** - 与 `ikelifetime Libreswan` 参数对应 - 用于加密流量的密钥的有效期。
- **Phase2 Lifetime** - 与 `salifetime Libreswan` 参数对应 - 在过期前连接的特定实例应多久。

**注意：**为了安全起见，加密密钥应该不时地更改。

- 远程网络 与 `rightsubnet Libreswan` 参数对应 - 应该通过 VPN 访问的目标专用远程网络。

检查 缩减 字段以启用缩小字段。请注意，它只在 IKEv2 协商中有效。

- 启用碎片 - 对应于 fragmentation Libreswan 参数 - 是否允许 IKE 分段。有效值为 yes (默认) 或 no。
- **Enable Mobike** - 与 mobike Libreswan 参数对应 - 是否允许移动和多形协议 (MOBIKE、RFC 455) 启用连接来迁移其端点，而无需从头开始重启连接。这可用于在有线、无线或者移动数据连接之间进行切换的移动设备。值为 no (默认) 或 yes。

6.

选择 IPv4 菜单条目：

#### IPv4 方法

- **Automatic (DHCP)** - 如果您要连接的网络使用 DHCP 服务器来分配动态 IP 地址，请选择这个选项。
- **Link-Local Only** - 如果您要连接的网络没有 DHCP 服务器且您不想手动分配 IP 地址，请选择这个选项。随机地址将根据 [RFC 3927](#) 分配，带有前缀 169.254/16。
- 如果要手动分配 IP 地址，请手动选择这个选项。
- 在这个连接中禁用 disable- IPv4。

#### DNS

在 DNS 部分，当 Automatic 为 ON 时，将其切换到 OFF 以输入您要用逗号分开的 DNS 服务器的 IP 地址。

#### Routes

请注意，在 Routes 部分，当 Automatic 为 ON 时，会使用 DHCP 的路由，但您也可以添加额外的静态路由。当 OFF 时，只使用静态路由。

- **address** - 输入远程网络或主机的 IP 地址。
- **netmask** - 以上输入的 IP 地址的子网掩码或前缀长度。
- **gateway** - 上面输入的远程网络或主机的 网关的 IP 地址。
- **指标** - 网络成本，赋予此路由的首选值。数值越低，优先级越高。

**仅将此连接用于其网络上的资源**

选择这个复选框以防止连接成为默认路由。选择这个选项意味着只有特别用于路由的流量才会通过连接自动获得，或者手动输入到连接上。

7.

要在 VPN 连接中配置 IPv6 设置，请选择 IPv6 菜单条目：

#### IPv6 Method

- **自动** 选择此选项以使用 IPv6 无状态地址自动配置(SLAAC)根据硬件地址和路由器公告(RA)创建自动、无状态的配置。
- **Automatic, DHCP only** - 选择这个选项来不使用 RA，但从 DHCPv6 请求信息以创建有状态的配置。
- **Link-Local Only** - 如果您要连接的网络没有 DHCP 服务器且您不想手动分配 IP 地址，请选择这个选项。随机地址将根据 [RFC 4862](#) 分配，前缀为 FE80::0。
- **如果要手动分配 IP 地址，请手动选择这个选项。**

- 在这个连接中禁用 **disable- IPv6**。

请注意，**DNS, Routes, Use this connection only for resources on its network** 项是 IPv4 的常规设置。

8. 编辑完 VPN 连接后，点添加按钮自定义配置或应用按钮为现有配置保存它。
9. 将配置文件切换到 ON 以激活 VPN 连接。
10. 如果您在带有 DHCP 或无状态地址自动配置(SLAAC)的网络中使用此主机，则连接可能会受到重定向的影响。有关详情和缓解步骤，[请参阅将 VPN 连接分配给专用路由表，以防止连接绕过隧道。](#)

## 其他资源

- **nm-settings-libreswan(5)**

### 7.17. 使用 NM-CONNECTION-EDITOR 配置 VPN 连接

如果使用带有图形界面的 Red Hat Enterprise Linux，您可以在 nm-connection-editor 应用程序中配置 VPN 连接。

## 先决条件

- 已安装 NetworkManager-libreswan-gnome 软件包。
- 如果您配置了互联网密钥交换版本 2 (IKEv2) 连接：
  - 证书导入到 IPsec 网络安全服务 (NSS) 数据库中。
  - NSS 数据库中的证书 **nickname** 是已知的。

## 流程

1.

打开终端窗口，输入：

```
$ nm-connection-editor
```

2.

点击 + 按钮来添加一个新的连接。

3.

选择 **IPsec based VPN** 连接类型，然后点击 **Create**。

4.

在 **VPN** 选项卡中：

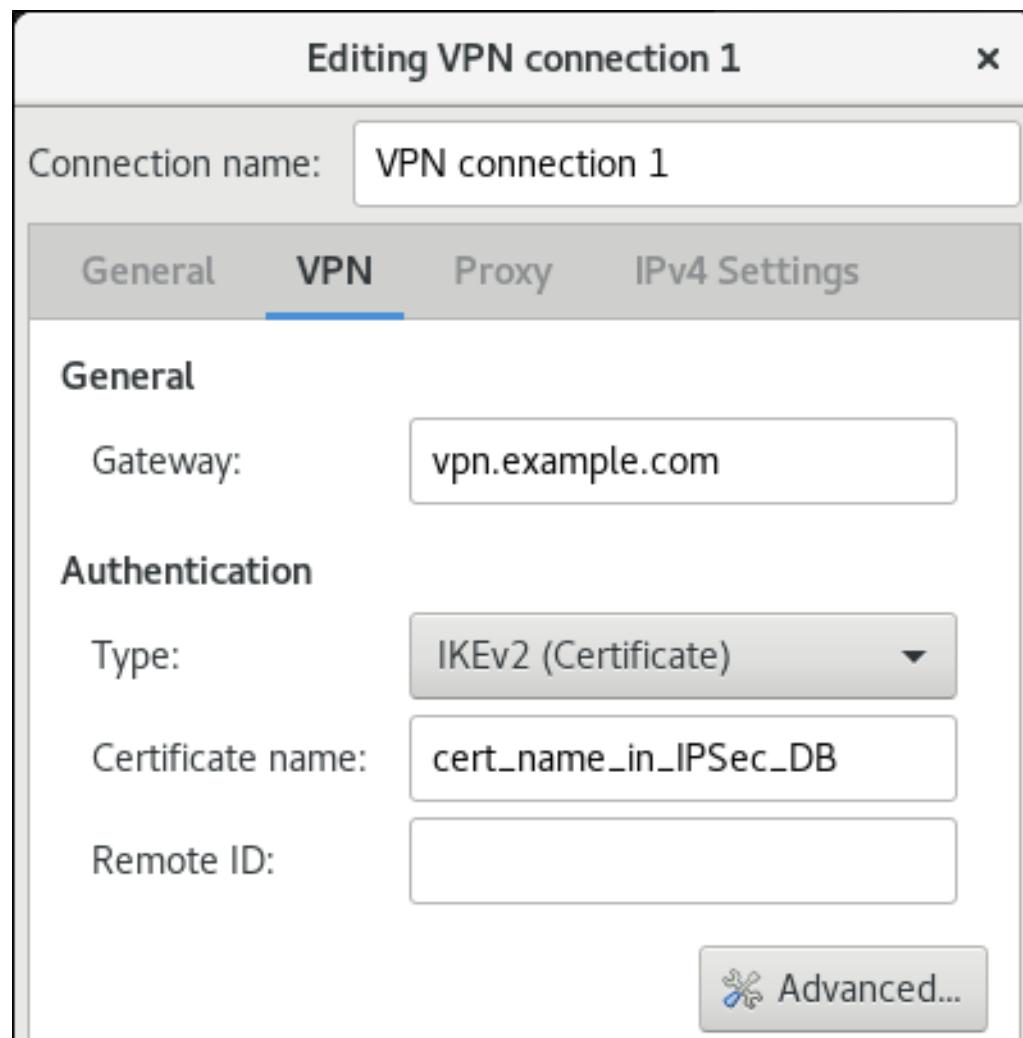
a.

在 **Gateway** 字段中输入 VPN 网关的主机名或 IP 地址，然后选择验证类型。根据验证类型，您必须输入不同的额外信息：

- **IKEv2 (认证)** 使用证书验证客户端，这会更安全。这个设置需要在 **IPsec NSS** 数据库中指定证书的 **nickname**
- **IKEv1(XAUTH)** 使用用户名和密码（预共享密钥）验证用户身份。此设置要求您输入以下值：
  - 用户名
  - 密码
  - 组名称
  - Secret

b.

如果远程服务器为 IKE 交换指定了本地标识符，在 **Remote ID** 字段中输入准确的字符串。在运行 **Libreswan** 的远程服务器中，这个值是在服务器的 **leftid** 参数中设置的。



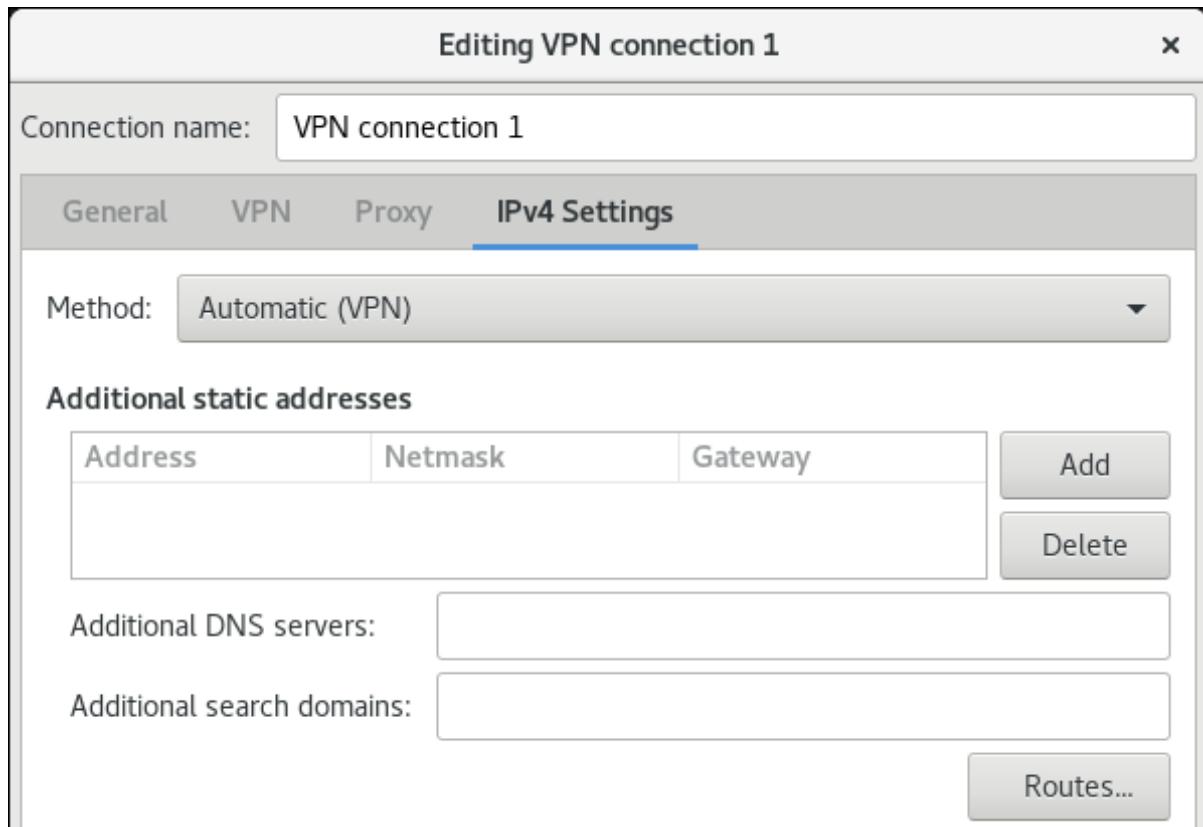
C.

可选：点 Advanced 按钮来配置其他设置。您可以配置以下设置：

- 身份识别
  - 域 -如果需要，请输入域名。
- 安全性
  - Phase1 Algorithms 对应于 ike Libreswan 参数。输入用来验证和设置加密频道的算法。

- Phase2 Algorithms 对应于 esp Libreswan 参数。输入用于 IPsec 协商的算法。

选择 Disable PFS 字段来关闭 Perfect Forward Secrecy(PFS)，以确保与不支持 PFS 的旧服务器兼容。
  - Phase1 Lifetime 与 ikelifetime Libreswan 参数对应。此参数定义用于加密流量的密钥的有效期。
  - Phase2 Lifetime 与 salifetime Libreswan 参数对应。这个参数定义安全关联有效期。
- 连接性
    - 远程网络 与 rightsubnet Libreswan 参数对应，并定义应通过 VPN 访问的目标专用远程网络。
- 检查 缩减 字段以启用缩小字段。请注意，它只在 IKEv2 协商中有效。
- Enable fragmentation 与 segmentation Libreswan 参数对应，并定义是否允许 IKE 分段。有效值为 yes (默认) 或 no。
  - Enable Mobike 与 mobike 参数对应。参数定义是否允许移动和多宿主协议 (MOBIKE) (RFC 4555) 启用连接来迁移其端点，而无需从头开始重启连接。这可用于在有线、无线或者移动数据连接之间进行切换的移动设备。值为 no (默认) 或 yes。
5. 在 IPv4 Settings 选项卡中，选择 IP 分配方法，并可选择设置额外的静态地址、DNS 服务器、搜索域和路由。



6. 保存连接。
7. 关闭 **nm-connection-editor**。
8. 如果您在带有 **DHCP** 或无状态地址自动配置(**SLAAC**)的网络中使用此主机，则连接可能会受到重定向的影响。有关详情和缓解步骤，[请参阅将 VPN 连接分配给专用路由表，以防止连接绕过隧道。](#)



### 注意

当您点 + 按钮添加新连接时， **NetworkManager** 会为那个连接创建新配置文件，然后打开同一个对话框来编辑现有连接。这两个对话框之间的区别在于现有连接配置集有详情菜单条目。

### 其他资源

- 您系统上的 **nm-settings-libreswan (5)** 手册页

### 7.18. 将 VPN 连接分配给专用路由表，以防止连接绕过隧道

DHCP 服务器和无状态地址自动配置(SLAAC)都可以将路由添加到客户端的路由表中。例如，恶意的 DHCP 服务器可以使用此功能强制带有 VPN 连接的主机通过物理接口而不是 VPN 隧道重定向流量。此漏洞也称为 TunnelVision，并参见 [CVE-2024-3661](#) 漏洞文章中所述。

要缓解此漏洞，您可以将 VPN 连接分配给专用路由表。这可防止 DHCP 配置或 SLAAC 来处理用于 VPN 隧道的网络数据包的路由决策。

如果至少有一个条件应用到您的环境，请按照以下步骤操作：

- 至少一个网络接口使用 DHCP 或 SLAAC。
- 您的网络不使用阻止恶意 DHCP 服务器的机制，如 DHCP 倾听。



#### 重要

通过 VPN 路由整个流量可防止主机访问本地网络资源。

#### 先决条件

- 您可以使用 NetworkManager 1.40.16-18 或更高版本。

#### 流程

1. 决定您要使用的路由表。以下步骤使用表 75。默认情况下，RHEL 不使用表 1-254，您可以使用其中任何一个。
2. 配置 VPN 连接配置文件，将 VPN 路由放在专用路由表中：

```
nmcli connection modify <vpn_connection_profile> ipv4.route-table 75 ipv6.route-table 75
```

3.

为您在上一命令中使用的表设置低优先级值：

```
nmcli connection modify <vpn_connection_profile> ipv4.routing-rules "priority 32345 from all table 75" ipv6.routing-rules "priority 32345 from all table 75"
```

优先级值可以是 1 到 32766 之间的任何值。值越低，优先级越高。

4.

重新连接 VPN 连接：

```
nmcli connection down <vpn_connection_profile>
nmcli connection up <vpn_connection_profile>
```

验证

1.

在表 75 中显示 IPv4 路由：

```
ip route show table 75
...
192.0.2.0/24 via 192.0.2.254 dev vpn_device proto static metric 50
default dev vpn_device proto static scope link metric 50
```

输出确认到远程网络和默认网关路由到路由表 75，因此所有流量都通过隧道路由。如果您在 VPN 连接配置集中设置 `ipv4.never-default true`，则不会创建默认路由，因此无法在此输出中可见。

2.

在表 75 中显示 IPv6 路由：

```
ip -6 route show table 75
...
2001:db8:1::/64 dev vpn_device proto kernel metric 50 pref medium
default dev vpn_device proto static metric 50 pref medium
```

输出确认到远程网络和默认网关路由到路由表 75，因此所有流量都通过隧道路由。如果您在 VPN 连接配置集中设置 `ipv4.never-default true`，则不会创建默认路由，因此无法在此输出中可见。

其他资源

- 

[CVE-2024-3661](#)

## 7.19. 其他资源

- 

[ipsec\(8\)](#)、[ipsec.conf\(5\)](#)、[ipsec.secrets\(5\)](#)、[ipsec\\_auto\(8\)](#) 和 [ipsec\\_rsasigkey\(8\)](#) 手册页。

- 

[/usr/share/doc/libreswan-版本/](#) 目录。

- 

[Libreswan 项目 Wiki](#)。

- 

[所有 Libreswan 手册页](#)。

- 

[NIST 特殊发布 800-77 : IPsec VPN 指南](#)。

## 第 8 章 配置 IP 隧道

与 VPN 类似，IP 隧道通过第三方网络（如互联网）直接连接两个网络。然而，不是所有的隧道协议都支持加密。

两个建立隧道网络的路由器至少需要两个接口：

- 一个连接到本地网络的接口
- 一个连接到建立隧道的网络的接口。

要建立隧道，您可以在两个路由器中使用来自远程子网的 IP 地址创建一个虚拟接口。

NetworkManager 支持以下 IP 隧道：

- 通用路由封装 (GRE)
- IPv6 上的通用路由封装 (IP6GRE)
- 通用路由封装终端接入点 (GRETAP)
- 通用路由登录在 IPv6 (IP6GRETAP) 上
- IPv4 over IPv4 (IPIP)
- IPv4 over IPv6 (IPIP6)
- IPv6 over IPv6 (IP6IP6)

- 

### 简单的互联网转换 (SIT)

根据类型，这些通道在 Open Systems Interconnection (OSI) 的第 2 层或 3 层动作。

#### 8.1. 配置 IPIP 隧道来封装 IPv4 数据包中的 IPV4 流量

IP over IP (IPIP) 隧道在 OSI 层 3 上运行，并封装 IPv4 数据包中的 IPv4 流量，如 RFC 2003 所述。

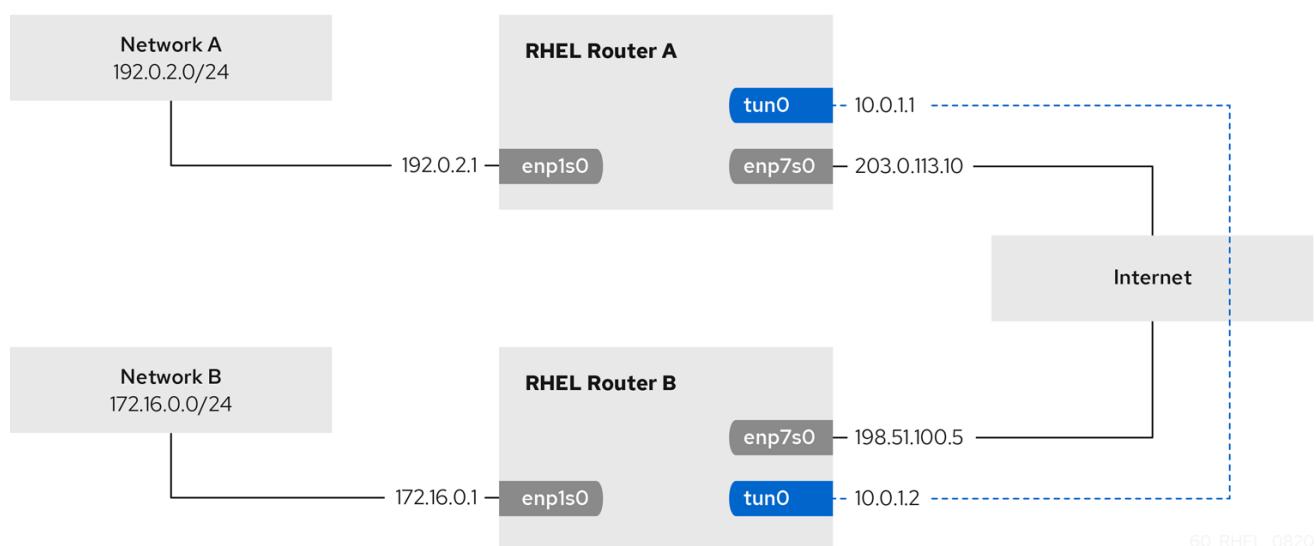


#### 重要

通过 IPIP 隧道发送的数据没有加密。出于安全考虑，只在已经加密的数据中使用隧道，比如 HTTPS。

请注意，IPIP 隧道只支持单播数据包。如果您需要支持多播的 IPv4 隧道，请参阅 [配置 GRE 隧道来封装 IPv4 数据包中的第 3 层流量](#)。

例如，您可以在两个 RHEL 路由器之间创建一个 IPIP 隧道来通过互联网连接两个内部子网，如下图所示：



#### 先决条件

- 每个 RHEL 路由器都有一个网络接口，它连接到其本地子网。
- 每个 RHEL 路由器都有一个网络接口，它连接到互联网。
- 您需要通过隧道发送的流量是 IPv4 单播。

## 流程

1. 在网络 A 的 RHEL 路由器上：

- a. 创建名为 tun0 的 IPIP 隧道接口：

```
nmcli connection add type ip-tunnel ip-tunnel.mode ipip con-name tun0 ifname tun0 remote 198.51.100.5 local 203.0.113.10
```

remote 和 local 参数设置远程和本地路由器的公共 IP 地址。

- b. 将 IPv4 地址设为 tun0 设备：

```
nmcli connection modify tun0 ipv4.addresses '10.0.1.1/30'
```

请注意，具有两个可用 IP 地址的 /30 子网足以满足隧道的需要。

- c. 将 tun0 连接配置为使用手动 IPv4 配置：

```
nmcli connection modify tun0 ipv4.method manual
```

- d. 添加一个静态路由，其将到 172.16.0.0/24 网络的流量路由到路由器 B 上的隧道 IP：

```
nmcli connection modify tun0 +ipv4.routes "172.16.0.0/24 10.0.1.2"
```

e.

启用 tun0 连接。

```
nmcli connection up tun0
```

f.

启用数据包转发：

```
echo "net.ipv4.ip_forward=1" > /etc/sysctl.d/95-IPv4-forwarding.conf
sysctl -p /etc/sysctl.d/95-IPv4-forwarding.conf
```

2.

在网络 B 中的 RHEL 路由器中：

a.

创建名为 tun0 的 IPIP 隧道接口：

```
nmcli connection add type ip-tunnel ip-tunnel.mode ipip con-name tun0 ifname
tun0 remote 203.0.113.10 local 198.51.100.5
```

remote 和 local 参数设置远程和本地路由器的公共 IP 地址。

b.

将 IPv4 地址设为 tun0 设备：

```
nmcli connection modify tun0 ipv4.addresses '10.0.1.2/30'
```

c.

将 tun0 连接配置为使用手动 IPv4 配置：

```
nmcli connection modify tun0 ipv4.method manual
```

d.

添加一个静态路由，其将路由到 192.0.2.0/24 网络的流量路由到路由器 A 上的隧道 IP：

```
nmcli connection modify tun0 +ipv4.routes "192.0.2.0/24 10.0.1.1"
```

e.

启用 tun0 连接。

```
nmcli connection up tun0
```

f.

启用数据包转发：

```
echo "net.ipv4.ip_forward=1" > /etc/sysctl.d/95-IPv4-forwarding.conf
sysctl -p /etc/sysctl.d/95-IPv4-forwarding.conf
```

验证

- 从每个 RHEL 路由器中， ping 路由器的内部接口的 IP 地址：

a.

在路由器 A 上， ping 172.16.0.1：

```
ping 172.16.0.1
```

b.

在路由器 B 上， ping 192.0.2.1：

```
ping 192.0.2.1
```

## 8.2. 配置 GRE 隧道来封装 IPv4 数据包中的第 3 层流量

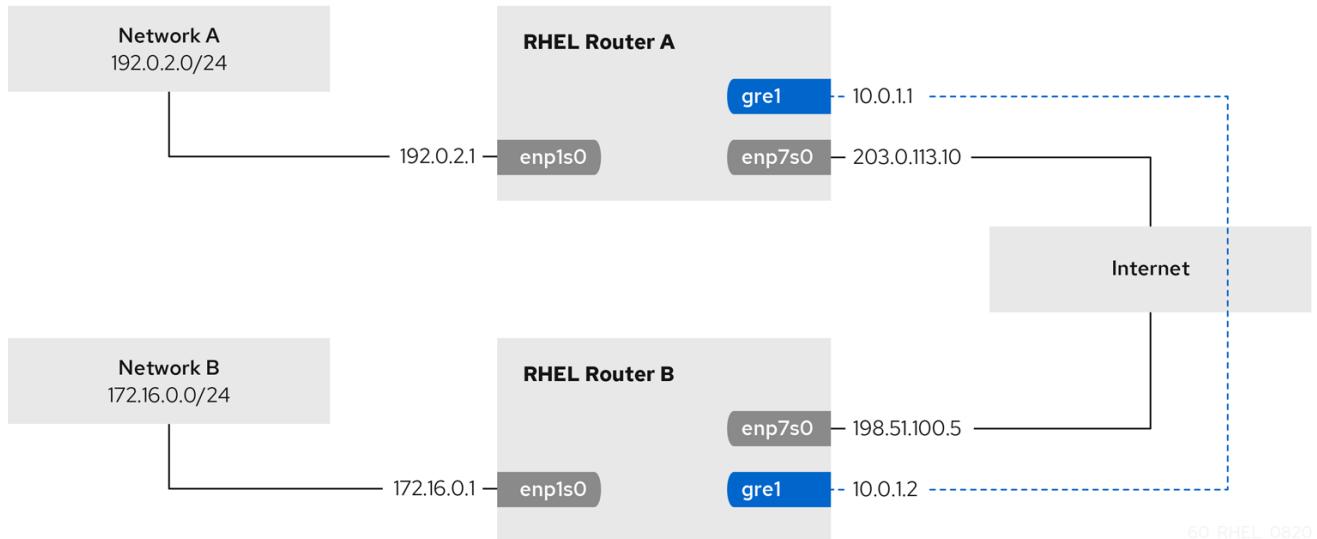
Generic Routing Encapsulation (GRE) 隧道封装 IPv4 数据包中的第 3 层流量，如 [RFC 2784](#) 所述。GRE 隧道可以使用有效的以太网类型封装任何第 3 层协议。



重要

通过 GRE 隧道发送的数据没有加密。出于安全考虑，只在已经加密的数据中使用隧道，比如 HTTPS。

例如，您可以在两个 RHEL 路由器之间创建一个 GRE 隧道来通过互联网连接两个内部子网，如下图所示：



## 先决条件

- 每个 RHEL 路由器都有一个网络接口，它连接到其本地子网。
- 每个 RHEL 路由器都有一个网络接口，它连接到互联网。

## 流程

1.

在网络 A 的 RHEL 路由器上：

a.

创建名为 gre1 的 GRE 隧道接口：

```
nmcli connection add type ip-tunnel ip-tunnel.mode gre con-name gre1 ifname
gre1 remote 198.51.100.5 local 203.0.113.10
```

**remote** 和 **local** 参数设置远程和本地路由器的公共 IP 地址。



**重要**

保留 gre0 设备名称。对该设备使用 gre1 或者不同的名称。

b.

将 IPv4 地址设为 gre1 设备：

```
nmcli connection modify gre1 ipv4.addresses '10.0.1.1/30'
```

请注意，具有两个可用 IP 地址的 /30 子网足以满足隧道的需要。

c.

将 gre1 连接配置为使用手动 IPv4 配置：

```
nmcli connection modify gre1 ipv4.method manual
```

d.

添加一个静态路由，其将到 172.16.0.0/24 网络的流量路由到路由器 B 上的隧道 IP：

```
nmcli connection modify gre1 +ipv4.routes "172.16.0.0/24 10.0.1.2"
```

e.

启用 gre1 连接。

```
nmcli connection up gre1
```

f.

启用数据包转发：

```
echo "net.ipv4.ip_forward=1" > /etc/sysctl.d/95-IPv4-forwarding.conf
sysctl -p /etc/sysctl.d/95-IPv4-forwarding.conf
```

2.

在网络 B 中的 RHEL 路由器中：

a.

创建名为 gre1 的 GRE 隧道接口：

```
nmcli connection add type ip-tunnel ip-tunnel.mode gre con-name gre1 ifname
gre1 remote 203.0.113.10 local 198.51.100.5
```

remote 和 local 参数设置远程和本地路由器的公共 IP 地址。

b.

将 IPv4 地址设为 gre1 设备：

```
nmcli connection modify gre1 ipv4.addresses '10.0.1.2/30'
```

c.

将 gre1 连接配置为使用手动 IPv4 配置：

```
nmcli connection modify gre1 ipv4.method manual
```

d.

添加一个静态路由，其将路由到 192.0.2.0/24 网络的流量路由到路由器 A 上的隧道 IP：

```
nmcli connection modify gre1 +ipv4.routes "192.0.2.0/24 10.0.1.1"
```

e.

启用 gre1 连接。

```
nmcli connection up gre1
```

f.

启用数据包转发：

```
echo "net.ipv4.ip_forward=1" > /etc/sysctl.d/95-IPv4-forwarding.conf
sysctl -p /etc/sysctl.d/95-IPv4-forwarding.conf
```

## 验证

1.

从每个 RHEL 路由器中，ping 路由器的内部接口的 IP 地址：

a.

在路由器 A 上，ping 172.16.0.1：

```
ping 172.16.0.1
```

b.

在路由器 B 上，ping 192.0.2.1：

```
ping 192.0.2.1
```

## 8.3. 配置 GRETAP 隧道来通过 IPV4 传输以太网帧

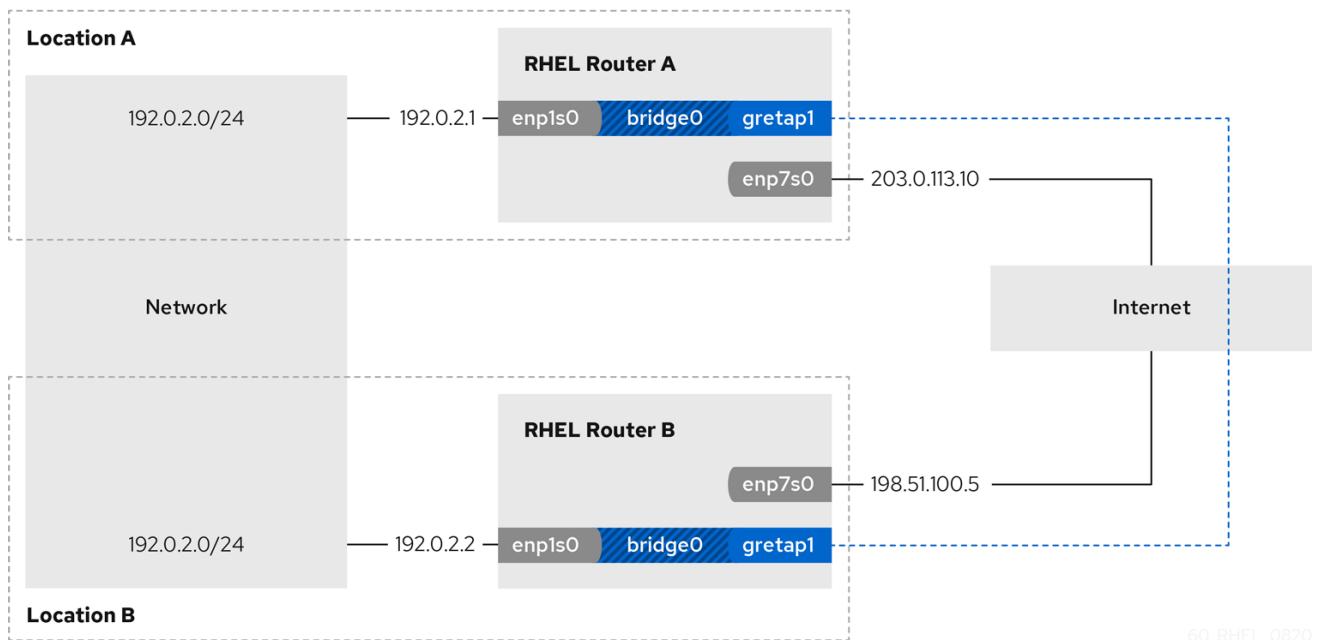
通用路由封装终端接入点(GRE)隧道在 OSI 级别 2 上运行，并封装 IPv4 数据包中的以太网流量，如 [RFC 2784](#) 所述。



### 重要

通过 GRE/TAP 隧道发送的数据没有加密。出于安全考虑，通过 VPN 或不同的加密连接建立隧道。

例如，您可以在两个 RHEL 路由器之间创建一个 GRE/TAP 隧道，以使用网桥连接两个网络，如下图所示：



### 先决条件

- 每个 RHEL 路由器都有一个网络接口，它连接到其本地网络，接口没有分配 IP 配置。
- 每个 RHEL 路由器都有一个网络接口，它连接到互联网。

### 流程

- 在网络 A 的 RHEL 路由器上：

a.

创建名为 **bridge0** 的网桥接口：

```
nmcli connection add type bridge con-name bridge0 ifname bridge0
```

b.

配置网桥的 IP 设置：

```
nmcli connection modify bridge0 ipv4.addresses '192.0.2.1/24'
nmcli connection modify bridge0 ipv4.method manual
```

c.

为连接到本地网络的接口添加新连接配置集到网桥：

```
nmcli connection add type ethernet slave-type bridge con-name bridge0-port1
ifname enp1s0 master bridge0
```

d.

为网桥添加 GRETAP 隧道接口的新连接配置集：

```
nmcli connection add type ip-tunnel ip-tunnel.mode gretap slave-type bridge
con-name bridge0-port2 ifname gretap1 remote 198.51.100.5 local 203.0.113.10
master bridge0
```

**remote** 和 **local** 参数设置远程和本地路由器的公共 IP 地址。



**重要**

保留 **gretap0** 设备名称。对该设备使用 **gretap1** 或者不同的名称。

e.

可选：如果您不需要，**STP (Spanning Tree Protocol)**：

```
nmcli connection modify bridge0 bridge.stp no
```

默认情况下，**STP** 被启用并导致在使用连接前出现延迟。

f.

配置激活 **bridge0** 连接会自动激活网桥端口：

```
nmcli connection modify bridge0 connection.autoconnect-slaves 1
```

g.

激活 **bridge0** 连接：

```
nmcli connection up bridge0
```

2.

在网络 B 中的 RHEL 路由器中：

a.

创建名为 **bridge0** 的网桥接口：

```
nmcli connection add type bridge con-name bridge0 ifname bridge0
```

b.

配置网桥的 IP 设置：

```
nmcli connection modify bridge0 ipv4.addresses '192.0.2.2/24'
nmcli connection modify bridge0 ipv4.method manual
```

c.

为连接到本地网络的接口添加新连接配置集到网桥：

```
nmcli connection add type ethernet slave-type bridge con-name bridge0-port1
ifname enp1s0 master bridge0
```

d.

为网桥添加 GRETAP 隧道接口的新连接配置集：

```
nmcli connection add type ip-tunnel ip-tunnel.mode gretap slave-type bridge
con-name bridge0-port2 ifname gretap1 remote 203.0.113.10 local 198.51.100.5
master bridge0
```

**remote** 和 **local** 参数设置远程和本地路由器的公共 IP 地址。

e.

可选：如果您不需要，**STP (Spanning Tree Protocol)**：

```
nmcli connection modify bridge0 bridge.stp no
```

t.

配置激活 bridge0 连接会自动激活网桥端口：

```
nmcli connection modify bridge0 connection.autoconnect-slaves 1
```

g.

激活 bridge0 连接：

```
nmcli connection up bridge0
```

验证

1.

在两个路由器上，验证 enp1s0 和 gretap1 连接是否已连接，并且 CONNECTION 列是否显示端口的连接名称：

```
nmcli device
nmcli device
DEVICE TYPE STATE CONNECTION
...
bridge0 bridge connected bridge0
enp1s0 ethernet connected bridge0-port1
gretap1 iptunnel connected bridge0-port2
```

2.

从每个 RHEL 路由器中，ping 路由器的内部接口的 IP 地址：

a.

在路由器 A 上，ping 192.0.2.2：

```
ping 192.0.2.2
```

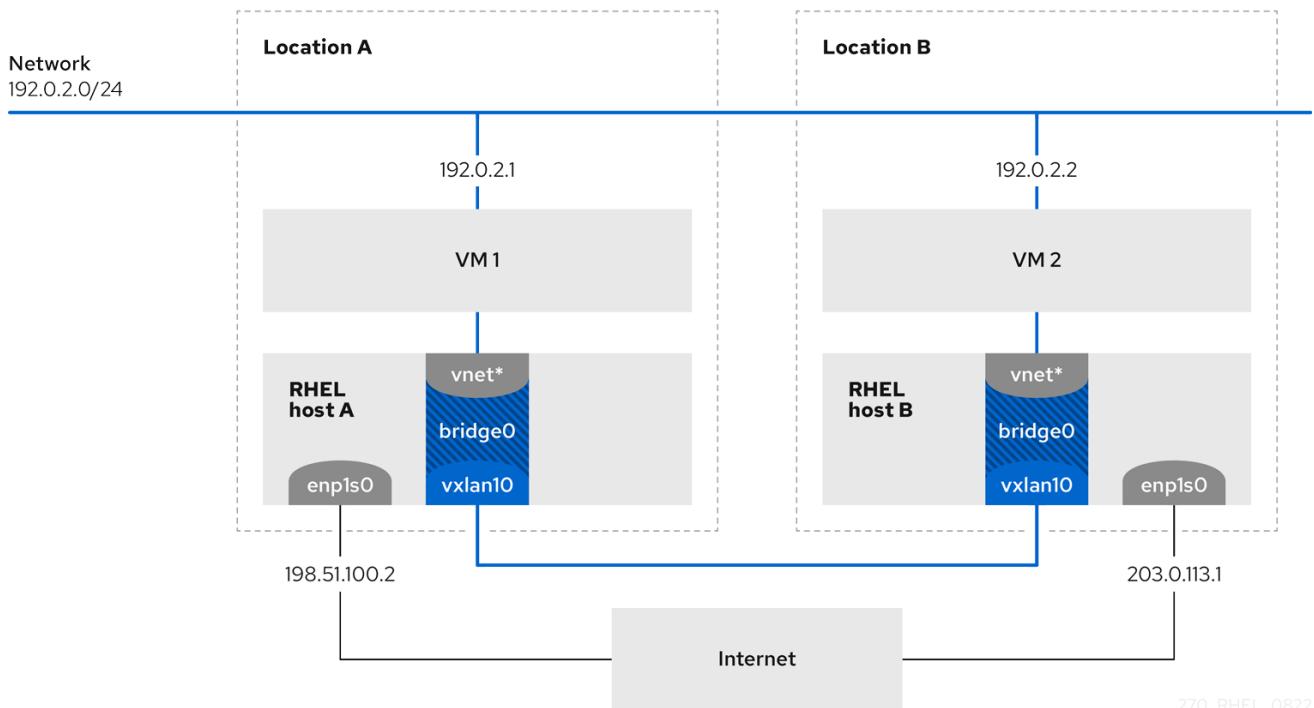
b.

在路由器 B 上，ping 192.0.2.1：

```
ping 192.0.2.1
```

## 第 9 章 使用 VXLAN 为虚拟机创建虚拟第 2 层域

虚拟可扩展局域网(VXLAN)是一种网络协议，它使用 UDP 协议在 IP 网络上对第 2 层流量进行隧道化。例如，在不同主机上运行的某些虚拟机(VM)可以通过 VXLAN 隧道进行通信。主机可以位于不同的子网中，甚至位于世界各地的不同数据中心。从虚拟机的角度来看，同一 VXLAN 中的其他虚拟机都在同一个第 2 层域中：



270\_RHEL\_0822

在本例中，RHEL-host-A 和 RHEL-host-B 使用网桥 br0 来在每台具有 VXLAN 为 vxlan10 的主机上连接虚拟机的虚拟网络。由于此配置，VXLAN 对虚拟机不可见，虚拟机不需要任何特殊的配置。如果您稍后将更多的虚拟机连接到同一虚拟网络，则虚拟机将自动成为同一虚拟第 2 层域的成员。



### 重要

就像正常的第 2 层流量一样，VXLAN 中的数据是不加密的。出于安全考虑，在 VPN 或其他类型的加密连接上使用 VXLAN。

#### 9.1. VXLAN 的优点

虚拟可扩展局域网(VXLAN)提供了以下主要优点：

- VXLAN 使用 24 位 ID。因此，您可以创建高达 16,777,216 个隔离网络。例如，虚拟 LAN(VLAN)只支持 4,096 个隔离网络。

- VXLAN 使用 IP 协议。这可让您路由流量，并在同一第 2 层域中的不同网络和位置虚拟运行系统。
- 与大多数隧道协议不同，VXLAN 不仅仅是一个点对点的网络。VXLAN 可以动态了解其他端点的 IP 地址，也可以使用静态配置的转发条目。
- 某些网卡支持 UDP 隧道相关的卸载功能。

## 其他资源

- `/usr/share/doc/kernel-doc-<kernel_version>/Documentation/networking/vxlan.rst` 由 kernel-doc 软件包提供

### 9.2. 在主机上配置以太网接口

要将 RHEL 虚拟机主机连接到以太网，请创建一个网络连接配置文件，配置 IP 设置，并激活配置文件。

在 RHEL 主机上运行此过程，并相应地调整 IP 地址配置。

## 先决条件

- 主机连接到以太网。

## 步骤

1. 在 NetworkManager 中添加新的以太网连接配置文件：

```
nmcli connection add con-name Example ifname enp1s0 type ethernet
```

2. 配置 IPv4 设置：

```
nmcli connection modify Example ipv4.addresses 198.51.100.2/24 ipv4.method
manual ipv4.gateway 198.51.100.254 ipv4.dns 198.51.100.200 ipv4.dns-search
example.com
```

如果网络使用 DHCP，**请跳过这一步。**

3.

**激活 Example 连接：**

```
nmcli connection up Example
```

**验证**

1.

**显示设备和连接的状态：**

```
nmcli device status
DEVICE TYPE STATE CONNECTION
enp1s0 ethernet connected Example
```

2.

**在远程网络中 ping 主机以验证 IP 设置：**

```
ping RHEL-host-B.example.com
```

请注意，在该主机上配置网络前，您无法 ping 其他虚拟机主机。

**其他资源**

- 

**您系统上的 nm-settings (5) 手册页**

### 9.3. 创建附加了 VXLAN 的网桥

要使虚拟可扩展局域网(VXLAN)对虚拟机(VM)不可见，请在主机上创建一个网桥，并将 VXLAN 附加给网桥。使用 NetworkManager 创建网桥和 VXLAN。您不能将虚拟机的任何流量访问点(TAP)设备（通常在主机上称为 vnet \*）添加到网桥。在虚拟机启动时，libvirthd 服务会动态添加它们。

在 RHEL 主机上运行此过程，并相应地调整 IP 地址。

## 步骤

1.

创建网桥 **br0** :

```
nmcli connection add type bridge con-name br0 ifname br0 ipv4.method disabled
ipv6.method disabled
```

此命令在网桥设备上设置没有 IPv4 和 IPv6 地址，因为此网桥在第 2 层工作。

2.

创建 VXLAN 接口，并将其附加到 **br0** :

```
nmcli connection add type vxlan slave-type bridge con-name br0-vxlan10 ifname
vxlan10 id 10 local 198.51.100.2 remote 203.0.113.1 master br0
```

这个命令使用以下设置：

- **ID 10** : 设置 VXLAN 标识符。
- **local 198.51.100.2** : 设置传出数据包的源 IP 地址。
- **remote 203.0.113.1** : 当目的地链路层地址在 VXLAN 设备转发数据库中未知时，设置要在传出数据包中使用的单播或多播IP地址。
- **Master br0** : 将要创建的此 VXLAN 连接设为 **br0** 连接中的一个端口。
- **ipv4.method disabled** 和 **ipv6.method disabled**: 在网桥上禁用 IPv4 和 IPv6。

默认情况下，NetworkManager 使用 8472 作为目的地端口。如果目的地端口不同，还要将 **destination-port <port\_number>** 选项传给命令。

3.

激活 **br0** 连接配置文件：

```
nmcli connection up br0
```

4.

在本地防火墙中为进入 UDP 连接打开端口 8472 :

```
firewall-cmd --permanent --add-port=8472/udp
firewall-cmd --reload
```

## 验证

- 

显示转发表 :

```
bridge fdb show dev vxlan10
2a:53:bd:d5:b3:0a master br0 permanent
00:00:00:00:00:00 dst 203.0.113.1 self permanent
...
```

## 其他资源

- 

您系统上的 [nm-settings \(5\)](#) 手册页

## 9.4. 在带有现有网桥的 LIBVIRT 中创建一个虚拟网络

要使虚拟机(VM)使用带有附加虚拟可扩展局域网(VXLAN) 的 br0 网桥，首先将虚拟网络添加到使用此网桥的 libvirtd 服务中。

### 先决条件

- 

您已安装了 libvirt 软件包。

- 

您已启动并启用了 libvirtd 服务。

- 

您已在 RHEL 上配置了带有 VXLAN 的 br0 设备。

### 步骤

1.

使用以下内容创建 ~/vxlan10-bridge.xml 文件：

```
<network>
<name>vxlan10-bridge</name>
<forward mode="bridge" />
<bridge name="br0" />
</network>
```

2.

使用 ~/vxlan10-bridge.xml 文件来在 libvirt 中创建一个新的虚拟网络：

```
virsh net-define ~/vxlan10-bridge.xml
```

3.

删除 ~/vxlan10-bridge.xml 文件：

```
rm ~/vxlan10-bridge.xml
```

4.

启动 vxlan10-bridge 虚拟网络：

```
virsh net-start vxlan10-bridge
```

5.

将 vxlan10-bridge 虚拟网络配置为在 libvirtd 服务启动时自动启动：

```
virsh net-autostart vxlan10-bridge
```

## 验证



显示虚拟网络列表：

```
virsh net-list
Name State Autostart Persistent

vxlan10-bridge active yes yes
...
```

## 其他资源



您系统上的 virsh (1) 手册页

## 9.5. 配置虚拟机以使用 VXLAN

要在主机上将虚拟机配置为使用带有附加虚拟可扩展 LAN(VXLAN)的网桥设备，请创建一个使用 **vxlan10-bridge** 虚拟网络的新虚拟机或更新现有虚拟机的设置以使用这个网络。

在 RHEL 主机上执行此流程。

### 先决条件

- 您在 libvirtd 中配置了 **vxlan10-bridge** 虚拟网络。

### 步骤

- 要创建新的虚拟机，并将其配置为使用 **vxlan10-bridge** 网络，请在创建虚拟机时将 **--network network:vxlan10-bridge** 选项传给 **virt-install** 命令：

```
virt-install ... --network network:vxlan10-bridge
```

- 要更改现有虚拟机的网络设置：

a.

将虚拟机的网络接口连接到 **vxlan10-bridge** 虚拟网络：

```
virt-xml VM_name --edit --network network=vxlan10-bridge
```

b.

关闭虚拟机，并重新启动它：

```
virsh shutdown VM_name
virsh start VM_name
```

### 验证

- 显示主机上虚拟机的虚拟网络接口：

```
virsh domiflist VM_name
Interface Type Source Model MAC

```

```
vnet1 bridge vxlan10-bridge virtio 52:54:00:c5:98:1c
```

2.

显示连接到 `vxlan10-bridge` 网桥的接口：

```
ip link show master vxlan10-bridge
18: vxlan10: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
 master br0 state UNKNOWN mode DEFAULT group default qlen 1000
 link/ether 2a:53:bd:d5:b3:0a brd ff:ff:ff:ff:ff:ff
19: vnet1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
 master br0 state UNKNOWN mode DEFAULT group default qlen 1000
 link/ether 52:54:00:c5:98:1c brd ff:ff:ff:ff:ff:ff
```

请注意，`libvirt` 服务会动态更新网桥的配置。当您启动使用了 `vxlan10-bridge` 网络的虚拟机时，主机上对应的 `vnet*` 设备会显示为网桥的端口。

3.

使用地址解析协议(ARP)请求来验证虚拟机是否在同一 VXLAN 中：

a.

启动同一 VXLAN 中的两个或多个虚拟机。

b.

将 ARP 请求从一个虚拟机发送到另一个虚拟机：

```
arping -c 1 192.0.2.2
ARPING 192.0.2.2 from 192.0.2.1 enp1s0
Unicast reply from 192.0.2.2 [52:54:00:c5:98:1c] 1.450ms
Sent 1 probe(s) (0 broadcast(s))
Received 1 response(s) (0 request(s), 0 broadcast(s))
```

如果命令显示回复，则虚拟机位于同一第 2 层域中，在这种情况下，是在同一 VXLAN 中。

安装 `iutils` 软件包以使用 `arping` 工具。

## 其他资源



您系统上的 `virt-install` (1) 和 `virt-xml` (1) 手册页



- 您系统上的 **virsh** (1) 和 **arping** (8) 手册页

## 第 10 章 管理 WIFI 连接

RHEL 提供多个实用程序和应用程序来配置和连接到 wifi 网络，例如：

- 使用 `nmcli` 工具，使用命令行配置连接。
- 使用 `nmtui` 应用程序在基于文本的用户界面中配置连接。
- 使用 GNOME 系统菜单快速连接到不需要任何配置的 wifi 网络。
- 使用 GNOME Settings 应用程序，使用 GNOME 应用程序配置连接。
- 使用 `nm-connection-editor` 应用程序在图形用户界面中配置连接。
- 使用 network RHEL 系统角色在一个或多个主机上自动化连接的配置。

### 10.1. 支持的 WIFI 安全类型

根据 wifi 网络支持的安全类型，您可以或多或少安全地传输数据。



警告

不要连接到不使用加密的网络，或者只支持不安全的 WEP 或 WPA 标准。

RHEL 8 支持以下 wifi 安全类型：

- **None** : 加密被禁用，数据在网络中以纯文本形式传输。
- **Enhanced Open** : 使用投机无线加密 (OWE) , 设备会协商唯一对主密钥 (PMK) 以在无线网络中加密连接，而无需身份验证。
- **WEP 40/128-bit Key (Hex or ASCII)**: 这个模式的 Wired Equivalent Privacy (WEP) 协议仅以十六进制或 ASCII 格式使用预共享密钥。WEP 已被弃用，并将在 RHEL 9.1 中被删除。
- **WEP 128 位密码**.这个模式的 WEP 协议使用密码短语的 MD5 哈希来生成 WEP 密钥。WEP 已被弃用，并将在 RHEL 9.1 中被删除。
- **动态 WEP (802.1x)** : 使用带动态键的 802.1X 和 EAP 的组合。WEP 已被弃用，并将在 RHEL 9.1 中被删除。
- **LEAP** : 由 Cisco 开发的轻量级可扩展验证协议，是可扩展身份验证协议 (EAP) 的专有版本。
- **WPA & WPA2 Personal** : 在个人模式中，WPA 和 Wi-Fi Protected Access 2 (WPA2) 验证方法使用预共享密钥。
- **WPA & WPA2 Enterprise** : 在企业模式中，WPA 和 WPA2 使用 EAP 框架，并对用户进行身份验证以远程身份验证服务 (RADIUS) 服务器。
- **WPA3 Personal - Wi-Fi Protected Access 3(WPA3) Personal** 使用 Simultaneous Authentication of Equals (SAE) 而不是预共享密钥 (PSK) 来防止字典攻击。WPA3 使用完美转发保密 (PFS)。

## 10.2. 使用 NMCLI 连接到 WIFI 网络

您可以使用 nmcli 工具连接到 wifi 网络。当您第一次尝试连接到网络时，实用程序会自动为其创建一个 NetworkManager 连接配置集。如果网络需要额外的设置，如静态 IP 地址，您可以在它自动创建后修改配置集。

## 先决条件

- 在主机上安装了 wifi 设备。
- wifi 设备已启用。要进行验证，请使用 `nmcli radio` 命令。

## 流程

1.

如果网络管理器 (NetworkManager) 中禁用了 wifi radio, 请启用此功能 :

```
nmcli radio wifi on
```

2.

可选：显示可用的 wifi 网络：

```
nmcli device wifi list
IN-USE BSSID SSID MODE CHAN RATE SIGNAL BARS SECURITY
00:53:00:2F:3B:08 Office Infra 44 270 Mbit/s 57 ████ WPA2 WPA3
00:53:00:15:03:BF -- Infra 1 130 Mbit/s 48 ████ WPA2 WPA3
```

服务设置标识符 (SSID) 列包含网络的名称。如果列显示 --，则此网络的接入点不会广播 SSID。

3.

连接到 wifi 网络：

```
nmcli device wifi connect Office --ask
Password: wifi-password
```

如果您希望在命令中设置密码而不是以交互方式输入密码，请在命令中使用 `password <wifi_password>` 选项，而不是`--ask`：

```
nmcli device wifi connect Office password <wifi_password>
```

请注意，如果网络需要静态 IP 地址，NetworkManager 无法在此时激活连接。您可以在后续步骤中配置 IP 地址。

4.

如果网络需要静态 IP 地址：

a.

配置 IPv4 地址设置，例如：

```
nmcli connection modify Office ipv4.method manual ipv4.addresses 192.0.2.1/24
ipv4.gateway 192.0.2.254 ipv4.dns 192.0.2.200 ipv4.dns-search example.com
```

b.

配置 IPv6 地址设置，例如：

```
nmcli connection modify Office ipv6.method manual ipv6.addresses
2001:db8:1::1/64 ipv6.gateway 2001:db8:1::fffe ipv6.dns 2001:db8:1::ffbb ipv6.dns-
search example.com
```

5.

重新激活连接：

```
nmcli connection up Office
```

## 验证

1.

显示活跃连接：

```
nmcli connection show --active
NAME ID TYPE DEVICE
Office 2501eb7e-7b16-4dc6-97ef-7cc460139a58 wifi wlp0s20f3
```

如果输出列出了您创建的 wifi 连接，则连接会活跃。

2.

Ping 主机名或 IP 地址：

```
ping -c 3 example.com
```

## 其他资源



您系统上的 **nm-settings-nmcli (5)** 手册页

### 10.3. 使用 GNOME 系统菜单连接到 WIFI 网络

您可以使用 GNOME 系统菜单连接到 wifi 网络。当您第一次连接到网络时，GNOME 会为它创建一个 NetworkManager 连接配置集。如果您将连接配置集配置为不自动连接，也可以使用 GNOME 系统菜单使用现有 NetworkManager 连接配置集手动连接到 wifi 网络。



#### 注意

第一次使用 GNOME 系统菜单建立与 wifi 网络的连接存在某种限制。例如，您无法配置 IP 地址设置。在这种情况下，首先配置连接：

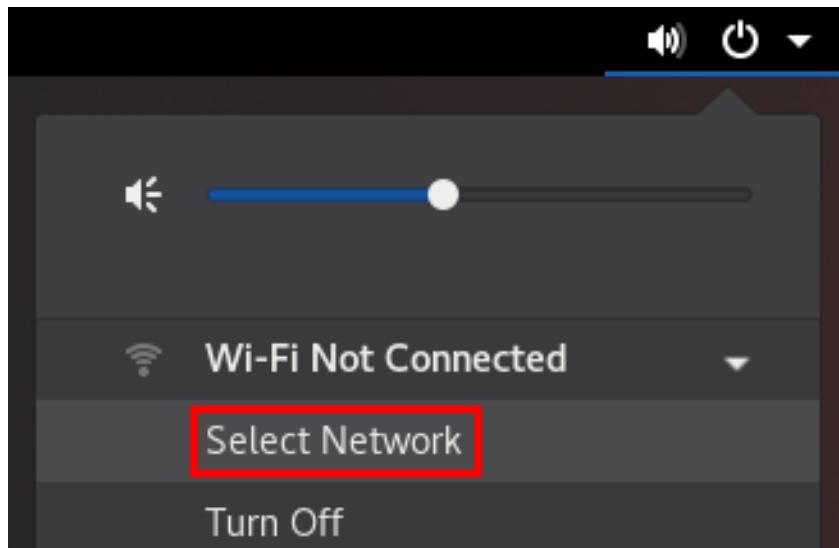
- 在 [GNOME settings](#) 应用程序中
- 在 [nm-connection-editor](#) 应用程序中
- 使用 [nmcli](#) 命令

#### 先决条件

- 在主机上安装了 wifi 设备。
- wifi 设备已启用。要进行验证，请使用 `nmcli radio` 命令。

#### 流程

1. 打开顶栏右侧的系统菜单。
2. 展开 **Wi-Fi Not Connected** 条目。
3. 点 **Select Network:**



4.

选择您要连接到的 wifi 网络。

5.

点 连接。

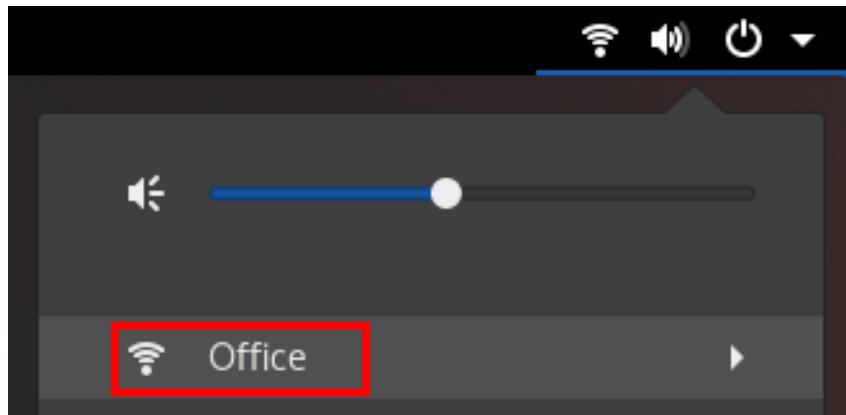
6.

如果这是您第一次连接到这个网络，请输入网络的密码，然后点 Connect。

## 验证

1.

打开顶栏右侧的系统菜单，验证 wifi 网络是否已连接：



如果网络出现在列表中，它已被连接。

2.

Ping 主机名或 IP 地址：

```
ping -c 3 example.com
```

## 10.4. 使用 GNOME 设置应用程序连接到 WIFI 网络

您可以使用名为 `gnome-control-center` 的 GNOME 设置 应用程序连接到 wifi 网络并配置连接。当您第一次连接到网络时，GNOME 会为它创建一个 NetworkManager 连接配置集。

在 GNOME 设置中，您可以为 RHEL 支持的所有 wifi 网络安全类型配置 wifi 连接。

### 先决条件

- 在主机上安装了 wifi 设备。
- wifi 设备已启用。要进行验证，请使用 `nmcli radio` 命令。

### 步骤

1. 按 Super 键，键入 Wi-Fi，然后按 Enter 键。
2. 点您要连接的 wifi 网络的名称。
3. 输入网络的密码，点 Connect。
4. 如果网络需要额外的设置，如静态 IP 地址或 WPA2 个人以外的安全类型：
  - a. 点网络名称旁边的齿轮图标。

- b. 可选：在 Details 标签页中配置网络配置集无法自动连接。

如果停用这个功能，您必须总是手动连接到网络，例如使用 GNOME settings 或 GNOME 系统菜单。

c.

在 IPv4 选项卡上配置 IPv4 设置，并在 IPv6 选项卡上配置 IPv6 设置。

d.

在 Security 选项卡中，选择网络验证，如 WPA3 Personal，然后输入密码。

根据所选安全性，应用程序会显示其他字段。相应地填充它们。详情请参阅 wifi 网络管理员。

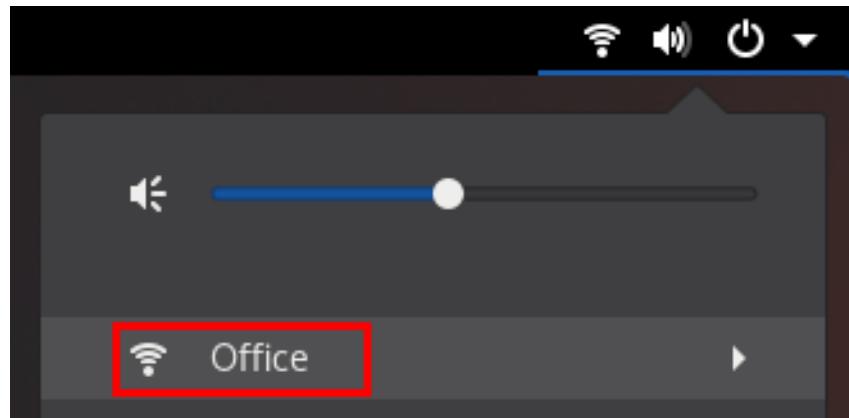
e.

点应用。

## 验证

1.

打开顶栏右侧的系统菜单，验证 wifi 网络是否已连接：



如果网络出现在列表中，它已被连接。

2.

Ping 主机名或 IP 地址：

```
ping -c 3 example.com
```

## 10.5. 使用 NMTUI 配置 WIFI 连接

nmtui 应用程序为 NetworkManager 提供了一个基于文本的用户界面。您可以使用 nmtui 连接到 wifi 网络。



## 注意

在 nmtui 中：

- 使用光标键导航。
- 选择一个按钮并按 Enter 键。
- 使用 空格 选择和清除复选框。
- 要返回上一个屏幕，请使用 ESC。

## 步骤

1.

启动 nmtui：

```
nmtui
```

2.

选择 Edit a connection，并按 Enter。

3.

按 Add 按钮。

4.

从网络类型列表中选择 Wi-Fi，然后按 Enter。

5.

可选：为要创建的 NetworkManager 配置集输入一个名称。

在有多个配置文件的主机上，有意义的名称可以更容易识别配置文件的用途。

6.

在 SSID 字段中输入 Wi-Fi 网络的名称，即服务集标识符(SSID)。

7.

将 **Mode** 字段设为默认值, **Client**。

8.

选择 **Security** 字段, 按 **Enter**, 然后从列表中设置网络的验证类型。

根据您选择的验证类型, nmtui 会显示不同的字段。

9.

填写与验证类型相关的字段。

10.

如果 Wi-Fi 网络需要静态 IP 地址 :

a.

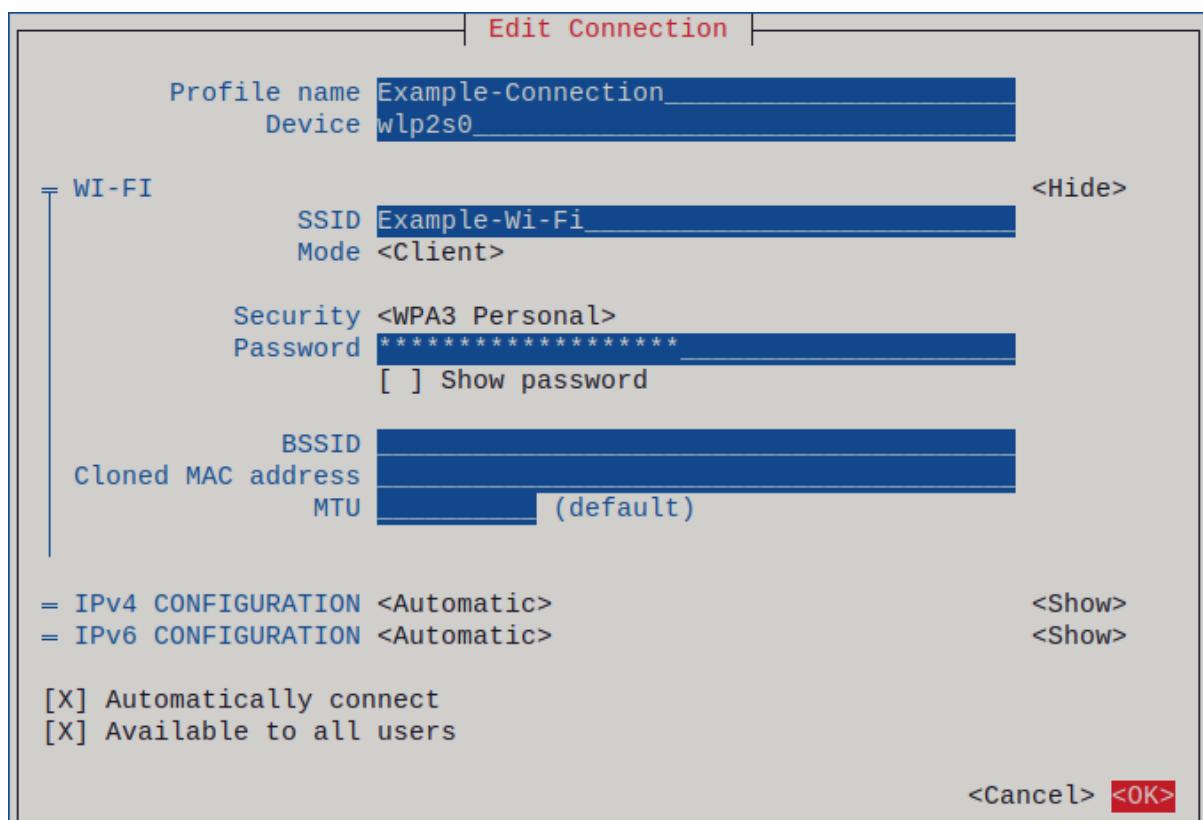
按协议旁边的 **Automatic** 按钮, 然后从显示的列表中选择 **Manual**。

b.

按您要配置的协议旁边的 **Show** 按钮, 来显示其他字段并填写它们。

11.

按 **OK** 按钮来创建并自动激活新连接。



12. 按 **Back** 按钮返回到主菜单。
  
13. 选择 **Quit**, 然后按 **Enter** 键关闭 **nmtui** 应用程序。

## 验证

1. 显示活跃连接：

```
nmcli connection show --active
NAME ID TYPE DEVICE
Office 2501eb7e-7b16-4dc6-97ef-7cc460139a58 wifi wlp0s20f3
```

如果输出列出了您创建的 **wifi** 连接，则连接会活跃。

2. Ping 主机名或 IP 地址：

```
ping -c 3 example.com
```

## 10.6. 使用 NM-CONNECTION-EDITOR 配置 WIFI 连接

您可以使用 **nm-connection-editor** 应用程序为无线网络创建连接配置集。在此应用程序中，您可以配置 RHEL 支持的所有 **wifi** 网络验证类型。

默认情况下，**NetworkManager** 为连接配置集启用自动连接功能，并在有可用时自动连接到保存的网络。

### 前提条件

- 在主机上安装了 **wifi** 设备。
  
- **wifi** 设备已启用。要进行验证，请使用 **nmcli radio** 命令。

### 流程

1. 打开终端窗口并输入：

```
nm-connection-editor
```

2. 点 + 按钮添加新连接。

3. 选择 Wi-Fi 连接类型，再点 Create。

4. 可选：为连接配置集设置名称。

5. 可选：在 General 选项卡中配置网络配置集无法自动连接。

如果停用这个功能，您必须总是手动连接到网络，例如使用 GNOME settings 或 GNOME 系统菜单。

6. 在 Wi-Fi 选项卡中，在 SSID 字段中输入服务设置标识符 (SSID)。

7. 在 Wi-Fi Security 选项卡中，为网络选择身份验证类型，如 WPA3 Personal，然后输入密码。

根据所选安全性，应用程序会显示其他字段。相应地填充它们。详情请参阅 wifi 网络管理员。

8. 在 IPv4 选项卡上配置 IPv4 设置，并在 IPv6 选项卡上配置 IPv6 设置。

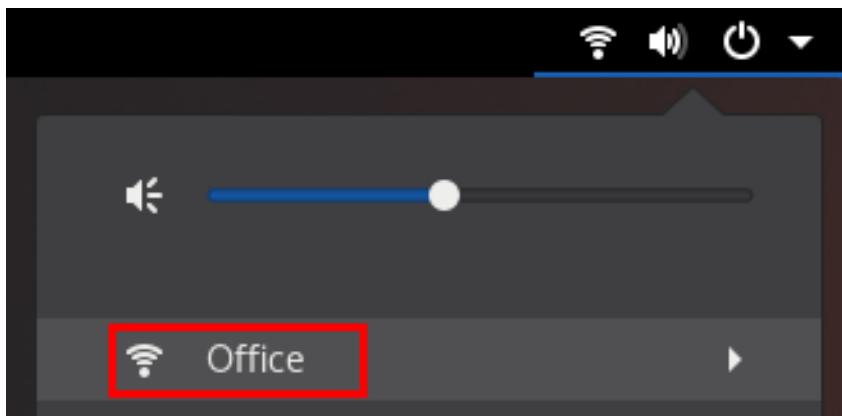
9. 点击 Save。

10. 关闭 Network Connections 窗口。

验证

1.

打开顶栏右侧的系统菜单，验证 wifi 网络是否已连接：



如果网络出现在列表中，它已被连接。

2.

Ping 主机名或 IP 地址：

```
ping -c 3 example.com
```

#### 10.7. 使用 NETWORK RHEL 系统角色配置带有 802.1X 网络身份验证的 WIFI 连接

网络访问控制(NAC)保护网络不受未授权客户端的访问。您可以在 NetworkManager 连接配置文件中指定身份验证所需的详情，以使客户端可以访问网络。通过使用 Ansible 和 network RHEL 系统角色，您可以自动化此过程，并在 playbook 中定义的主机上远程配置连接配置文件。

您可以使用 Ansible playbook 将私钥、证书和 CA 证书复制到客户端，然后使用 network RHEL 系统角色配置具有 802.1X 网络身份验证的连接配置文件。

#### 先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 sudo 权限。

- 网络支持 802.1X 网络身份验证。
- 您已在受管节点上安装了 `wpa_supplicant` 软件包。
- DHCP 位于受管节点的网络中。
- control 节点上存在 TLS 身份验证所需的以下文件：
  - 客户端密钥存储在 `/srv/data/client.key` 文件中。
  - 客户端证书存储在 `/srv/data/client.crt` 文件中。
  - CA 证书存储在 `/srv/data/ca.crt` 文件中。

## 流程

1. 将您的敏感变量存储在一个加密文件中：
  - a. 创建 vault：

```
$ ansible-vault create ~/vault.yml
New Vault password: <vault_password>
Confirm New Vault password: <vault_password>
```
  - b. 在 `ansible-vault create` 命令打开编辑器后，以 `<key>: <value>` 格式输入敏感数据：

```
pwd: <password>
```
  - c. 保存更改，并关闭编辑器。Ansible 加密 vault 中的数据。

2.

创建一个包含以下内容的 playbook 文件，如 ~/playbook.yml：

```

- name: Configure a wifi connection with 802.1X authentication
 hosts: managed-node-01.example.com
 tasks:
 - name: Copy client key for 802.1X authentication
 ansible.builtin.copy:
 src: "/srv/data/client.key"
 dest: "/etc/pki/tls/private/client.key"
 mode: 0400

 - name: Copy client certificate for 802.1X authentication
 ansible.builtin.copy:
 src: "/srv/data/client.crt"
 dest: "/etc/pki/tls/certs/client.crt"

 - name: Copy CA certificate for 802.1X authentication
 ansible.builtin.copy:
 src: "/srv/data/ca.crt"
 dest: "/etc/pki/ca-trust/source/anchors/ca.crt"

 - name: Wifi connection profile with dynamic IP address settings and 802.1X
 ansible.builtin.import_role:
 name: redhat.rhel_system_roles.network
 vars:
 network_connections:
 - name: Wifi connection profile with dynamic IP address settings and 802.1X
 interface_name: wlp1s0
 state: up
 type: wireless
 autoconnect: yes
 ip:
 dhcp4: true
 auto6: true
 wireless:
 ssid: "Example-wifi"
 key_mgmt: "wpa-eap"
 ieee802_1x:
 identity: <user_name>
 eap: tls
 private_key: "/etc/pki/tls/client.key"
 private_key_password: "{{ pwd }}"
 private_key_password_flags: none
 client_cert: "/etc/pki/tls/client.pem"
 ca_cert: "/etc/pki/tls/cacert.pem"
 domain_suffix_match: "example.com"
```

示例 playbook 中指定的设置包括以下内容：

**ieee802\_1x**

此变量包含与 802.1X 相关的设置。

#### eap: tls

将配置文件配置为用于可扩展身份验证协议(EAP)的基于证书的 TLS 身份验证方法。

有关 playbook 中使用的所有变量的详情，请查看控制节点上的  
`/usr/share/ansible/roles/rhel-system-roles.network/README.md` 文件。

3.

验证 playbook 语法：

```
$ ansible-playbook --ask-vault-pass --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

4.

运行 playbook：

```
$ ansible-playbook --ask-vault-pass ~/playbook.yml
```

### 其他资源

- 

`/usr/share/ansible/roles/rhel-system-roles.network/README.md` 文件

- 

`/usr/share/doc/rhel-system-roles/network/` 目录

## 10.8. 使用 NMCLI 在现有配置文件中配置带有 802.1X 网络身份验证的 WIFI 连接

使用 nmcli 工具，您可以将客户端配置为向网络进行身份验证。例如，您可以在名为 `wlp1s0` 的 NetworkManager wifi 连接配置文件中使用 Microsoft Challenge-Handshake Authentication Protocol 版本 2 (MSCHAPv2) 配置 Protected Extensible Authentication Protocol(PEAP) 身份验证。

### 前提条件

- 网络必须具有 802.1X 网络身份验证。
- Wi-Fi 连接配置集存在于 NetworkManager 中，且具有有效的 IP 配置。
- 如果需要客户端验证验证方的证书，则证书颁发机构(CA)证书必须存储在 /etc/pki/ca-trust/source/anchors/ 目录中。
- wpa\_supplicant 软件包已安装。

## 流程

1. 将 Wi-Fi 安全模式设为 wpa-eap、将可扩展验证协议(EAP)设为 peap，将内部验证协议设为 mschapv2，和用户名：

```
nmcli connection modify wlp1s0 wireless-security.key-mgmt wpa-eap 802-1x.eap
peap 802-1x.phase2-auth mschapv2 802-1x.identity user_name
```

请注意，您必须在单个命令中设置 wireless-security.key-mgmt、802-1x.eap、802-1x.phase2-auth 和 802-1x.identity 参数。

2. 可选：将密码存储在配置中：

```
nmcli connection modify wlp1s0 802-1x.password password
```

### 重要

默认情况下，NetworkManager 在 /etc/sysconfig/network-scripts/keys-connection\_name 文件中以纯文本形式存储密码，该文件只对 root 用户可读。但是，配置文件中的纯文本密码可能会造成安全风险。

要提高安全性，请将 802-1x.password-flags 参数设置为 agent-owned。使用这个设置，在具有 GNOME 桌面环境或运行 nm-applet 的服务器上，NetworkManager 会在您首先解锁密钥环后从这些服务检索密码。在其他情况下，NetworkManager 会提示输入密码。



3.

如果客户端需要验证验证器的证书，请将连接配置文件中的 `802-1x.ca-cert` 参数设为 CA 证书的路径：

```
nmcli connection modify wlp1s0 802-1x.ca-cert /etc/pki/ca-trust/source/anchors/ca.crt
```



#### 注意

为安全起见，客户端应验证认证方的证书。

4.

激活连接配置集：

```
nmcli connection up wlp1s0
```

验证

- 访问需要网络身份验证的网络上的资源。

#### 其他资源

- [管理 wifi 连接](#)
- 您系统上的 `nm-settings` (5) 和 `nmcli` (1) 手册页

#### 10.9. 手动设置无线规范域

在 RHEL 上，`udev` 规则执行 `setregdomain` 工具来设置无线规范域。然后，实用程序为内核提供此信息。

默认情况下，`setregdomain` 会尝试自动决定国家代码。如果此操作失败，则无线规范域设置可能会出错。要临时解决这个问题，您可以手动设置国家代码。



## 重要

手动设置规范域将禁用自动检测。因此，如果您稍后在不同的国家/地区使用计算机，之前配置的设置可能不再正确。在这种情况下，删除 /etc/sysconfig/regdomain 文件以切回到自动检测，或使用此流程再次更新规范域设置。

## 先决条件

- wifi 设备的驱动程序支持更改规范域。

## 流程

1. 可选：显示当前规范域设置：

```
iw reg get
global
country US: DFS-FCC
...
```

2. 使用以下内容创建 /etc/sysconfig/regdomain 文件：

```
COUNTRY=<country_code>
```

将 COUNTRY 变量设置为一个 ISO 3166-1 alpha2 国家代码，如 DE 代表德国，US 代表美国。

3. 设置规范域：

```
setregdomain
```

## 验证

- 显示规范域设置：

```
iw reg get
global
country DE: DFS-ETSI
...
```

## 其他资源

- 您系统上的 **iw** (8), **setregdomain** (1) 和 **regulatory.bin** (5) 手册页
- [ISO 3166 国家代码](#)

## 第 11 章 将 RHEL 配置为 WPA2 或 WPA3 个人访问令牌

在具有 wifi 设备的主机上，您可以使用 NetworkManager 将这个主机配置为接入点。Wi-Fi Protected Access 2 (WPA2) 和 Wi-Fi Protected Access 3 (WPA3) 提供安全验证方法，无线客户端可以使用预共享密钥(PSK)连接到访问点，并使用 RHEL 主机上和网络中的服务。

当您配置接入点时，NetworkManager 会自动：

- 配置 dnsmasq 服务来为客户端提供 DHCP 和 DNS 服务
- 启用 IP 转发
- 添加 nftables 防火墙规则来伪装来自 wifi 设备的流量并配置 IP 转发

### 先决条件

- wifi 设备支持在接入点模式下运行。
- wifi 设备没有使用。
- 主机可以访问互联网。

### 流程

1. 列出 wifi 设备来识别应提供访问点的 wifi 设备：

```
nmcli device status | grep wifi
wlp0s20f3 wifi disconnected --
```

2. 验证该设备是否支持访问点模式：

```
nmcli -f WIFI-PROPERTIES.AP device show wlp0s20f3
WIFI-PROPERTIES.AP: yes
```

要使用 wifi 设备作为接入点，该设备必须支持此功能。

3.

安装 dnsmasq 和 NetworkManager-wifi 软件包：

```
yum install dnsmasq NetworkManager-wifi
```

NetworkManager 使用 dnsmasq 服务为访问点的客户端提供 DHCP 和 DNS 服务。

4.

创建初始访问点配置：

```
nmcli device wifi hotspot ifname wlp0s20f3 con-name Example-Hotspot ssid Example-Hotspot password "password"
```

此命令在 wlp0s20f3 设备（提供 WPA2 和 WPA3 个人访问令牌）上创建一个连接配置集。无线网络的名称，Service Set Identifier (SSID) 是 Example-Hotspot，并使用预共享密钥 密码。

5.

可选：将访问点配置为只支持 WPA3：

```
nmcli connection modify Example-Hotspot 802-11-wireless-security.key-mgmt sae
```

6.

默认情况下，NetworkManager 使用 IP 地址 10.42.0.1 作为 wifi 设备，并将剩余的 10.42.0.0/24 子网的 IP 地址分配给客户端。要配置不同的子网和 IP 地址，请输入：

```
nmcli connection modify Example-Hotspot ipv4.addresses 192.0.2.254/24
```

您设置的 IP 地址（本例中为 192.0.2.254）是 NetworkManager 分配给 wifi 设备的 IP 地址。客户端将此 IP 地址用作默认网关和 DNS 服务器。

7.

激活连接配置集：

```
nmcli connection up Example-Hotspot
```

验证

1.

在服务器中：

a.

验证 NetworkManager 是否启动了 dnsmasq 服务，并且服务侦听端口 67 (DHCP) 和 53 (DNS)：

```
ss -tulpn | grep -E ":53|:67"
udp UNCONN 0 0 10.42.0.1:53 0.0.0.0:* users:(("dnsmasq",pid=55905,fd=6))
udp UNCONN 0 0 0.0.0.0:67 0.0.0.0:* users:(("dnsmasq",pid=55905,fd=4))
tcp LISTEN 0 32 10.42.0.1:53 0.0.0.0:* users:(("dnsmasq",pid=55905,fd=7))
```

b.

显示 nftables 规则集，以确保 NetworkManager 为来自 10.42.0.0/24 子网的流量启用转发和伪装：

```
nft list ruleset
table ip nm-shared-wlp0s20f3 {
 chain nat_postrouting {
 type nat hook postrouting priority srcnat; policy accept;
 ip saddr 10.42.0.0/24 ip daddr != 10.42.0.0/24 masquerade
 }

 chain filter_forward {
 type filter hook forward priority filter; policy accept;
 ip daddr 10.42.0.0/24 oifname "wlp0s20f3" ct state { established, related } accept
 ip saddr 10.42.0.0/24 iifname "wlp0s20f3" accept
 iifname "wlp0s20f3" oifname "wlp0s20f3" accept
 iifname "wlp0s20f3" reject
 oifname "wlp0s20f3" reject
 }
}
```

2.

在带有 wifi 适配器的客户端中：

a.

显示可用网络列表：

```
nmcli device wifi
IN-USE BSSID SSID MODE CHAN RATE SIGNAL BARS
SECURITY
00:53:00:88:29:04 Example-Hotspot Infra 11 130 Mbit/s 62 ██████████
WPA3
...
```

b.

连接到 Example-Hotspot 无线网络。请参阅[管理 Wi-Fi 连接](#)。

c.

对远程网络或互联网中的主机发出 ping 以验证连接是否正常工作：

```
ping -c 3 www.redhat.com
```

## 其他资源



您系统上的 **nm-settings (5)** 手册页

## 第 12 章 使用 MACSEC 加密同一物理网络中的第 2 层流量

您可以使用 **MACsec** 来保护两个设备（点到点）之间的通信。例如，您的分支办公室通过城际以太网与中心办公室连接，您可以在连接办公室的两个主机上配置 **MACsec**，以提高安全性。

### 12.1. MACSEC 如何提高安全性

介质访问控制安全(**MACsec**)是一种第 2 层协议，它保护以太网链路上不同的流量类型，包括：

- 动态主机配置协议(**DHCP**)
- 地址解析协议(**ARP**)
- IPv4 和 IPv6 流量
- 任何通过 IP 的流量，如 **TCP** 或 **UDP**

**MACsec** 默认使用 **GCM-AES-128** 算法加密并验证 LAN 中的所有流量，并使用预共享密钥在参与的主机之间建立连接。要更改预共享密钥，您必须更新所有使用 **MACsec** 的网络主机上的 NM 配置。

**MACsec** 连接使用一个以太网设备，如以太网网卡、**VLAN** 或隧道设备作为父设备。您只能在 **MACsec** 设备上设置 IP 配置，以便只使用加密连接与其他主机进行通信，或者在父设备上设置 IP 配置。在后者的情况下，您可以使用父设备使用未加密连接和 **MACsec** 设备加密连接与其他主机通信。

**macsec** 不需要任何特殊硬件。例如，您可以使用任何交换机，除非您只想在主机和交换机之间加密流量。在这种情况下，交换机还必须支持 **MACsec**。

换句话说，您可以为两种常见场景配置 **MACsec**：

- **host-to-host**

- **host-to-switch 和 switch-to-other-hosts**

**重要**

您只能在位于同一物理或虚拟 LAN 的主机之间使用 MACsec。

使用 MACsec 安全标准保护链路层的通信，也称为 Open Systems Interconnection (OSI) 模型的第 2 层，提供以下显著优点：

- 第 2 层的加密消除了在第 7 层加密单个服务的需要。这减少了管理与每个主机上每个端点的大量证书关联的开销。
- 直接连接的网络设备（如路由器和交换机）之间的点对点安全性。
- 不需要对应用程序和高层协议进行更改。

## 其他资源

- [MACsec：加密网络流量的一个不同的解决方案](#)

## 12.2. 使用 NMCLI 配置 MACSEC 连接

您可以使用 nmcli 工具将以太网接口配置为使用 MACsec。例如，您可以在通过以太网连接的两个主机之间创建一个 MACsec 连接。

### 流程

1. 在配置 MACsec 的第一个主机上：
  - 为预共享密钥创建连接关联密钥(CKA)和连接关联密钥名称(CKN)：

a.

创建一个 16 字节的十六进制 CAK :

```
dd if=/dev/urandom count=16 bs=1 2> /dev/null | hexdump -e '1/2 "%04x"'
50b71a8ef0bd5751ea76de6d6c98c03a
```

b.

创建一个 32 字节的十六进制 CKN :

```
dd if=/dev/urandom count=32 bs=1 2> /dev/null | hexdump -e '1/2 "%04x"'
f2b4297d39da7330910a74abc0449feb45b5c0b9fc23df1430e1898fcf1c4550
```

2.

在您要通过 MACsec 连接连接的两个主机上：

3.

创建 MACsec 连接：

```
nmcli connection add type macsec con-name macsec0 ifname macsec0
connection.autoconnect yes macsec.parent enp1s0 macsec.mode psk macsec.mka-
cak 50b71a8ef0bd5751ea76de6d6c98c03a macsec.mka-ckn
f2b4297d39da7330910a74abc0449feb45b5c0b9fc23df1430e1898fcf1c4550
```

在 macsec.mka-cak 和 macsec.mka-ckn 参数中使用上一步生成的 CAK 和 CKN。在 MACsec-protected 网络的每个主机上，这些值必须相同。

4.

配置 MACsec 连接中的 IP 设置。

a.

配置 IPv4 设置。例如，要为 macsec0 连接设置静态 IPv4 地址、网络掩码、默认网关和 DNS 服务器，请输入：

```
nmcli connection modify macsec0 ipv4.method manual ipv4.addresses
'192.0.2.1/24' ipv4.gateway '192.0.2.254' ipv4.dns '192.0.2.253'
```

b.

配置 IPv6 设置。例如，要为 macsec0 连接设置静态 IPv6 地址、网络掩码、默认网关和 DNS 服务器，请输入：

```
nmcli connection modify macsec0 ipv6.method manual ipv6.addresses
'2001:db8:1::1/32' ipv6.gateway '2001:db8:1::ffffe' ipv6.dns '2001:db8:1::ffffd'
```

5.

激活连接：

```
nmcli connection up macsec0
```

验证

1.

验证流量是否加密：

```
tcpdump -nn -i enp1s0
```

2.

可选：显示未加密的流量：

```
tcpdump -nn -i macsec0
```

3.

显示 **MACsec** 统计信息：

```
ip macsec show
```

4.

显示每种保护类型的单独的计数器：仅完整性（关闭加密）和加密（打开加密）

```
ip -s macsec show
```

其他资源



[MACsec：加密网络流量的一个不同的解决方案](#)

## 第 13 章 开始使用 IPVLAN

IPVLAN 是虚拟网络设备的驱动程序，可在容器环境中用于访问主机网络。IPVLAN 会将一个 MAC 地址公开给外部网络，而不管主机网络中所创建的 IPVLAN 设备的数量。这意味着，用户可以在多个容器中有多个 IPVLAN 设备，相应的交换机会读取单个 MAC 地址。当本地交换机对它可管理的 MAC 地址的总数施加约束时，IPVLAN 驱动程序很有用。

### 13.1. IPVLAN 模式

IPVLAN 有以下模式可用：

- L2 模式

在 IPVLAN L2 模式 中，虚拟设备接收并响应地址解析协议(ARP)请求。netfilter 框架仅在拥有虚拟设备的容器中运行。容器化流量的默认命名空间中不会执行 netfilter 链。使用L2 模式会提供良好的性能，但对网络流量的控制要小。

- L3 模式

在 L3 模式 中，虚拟设备只处理 L3 以上的流量。虚拟设备不响应 ARP 请求，用户必须手动为相关点上的 IPVLAN IP 地址配置邻居条目。相关容器的出口流量位于默认命名空间中的 netfilter POSTROUTING 和 OUTPUT 链上，而入口流量以与 L2 模式相同的方式被线程化。使用L3 模式会提供很好的控制，但可能会降低网络流量性能。

- L3S 模式

在 L3S 模式 中，虚拟设备处理方式与 L3 模式 中的处理方式相同，但相关容器的出口和入口流量都位于默认命名空间中的 netfilter 链上。L3S 模式 的行为方式和 L3 模式 相似，但提供了对网络的更大控制。



#### 注意

对于 L3 和 L3S 模式，IPVLAN 虚拟设备不接收广播和多播流量。

### 13.2. IPVLAN 和 MACVLAN 的比较

下表显示了 MACVLAN 和 IPVLAN 之间的主要区别：

MACVLAN	IPVLAN
为每个 MACVLAN 设备使用 MAC 地址。  请注意，如果交换机达到了其 MAC 表中可以存储的最大 MAC 地址数，则连接可能会丢失。	使用不限制 IPVLAN 设备数的单个 MAC 地址。
全局命名空间的 Netfilter 规则不会影响子命名空间中到 MACVLAN 设备或来自 MACVLAN 设备的流量。	可以在 L3 mode L3S mode 下控制到 IPVLAN 设备或来自 IPVLAN 设备的流量。

IPVLAN 和 MACVLAN 不需要任何级别的封装。

### 13.3. 使用 iproute2 创建和配置 IPVLAN 设备

此流程演示了如何使用 iproute2 设置 IPVLAN 设备。

#### 流程

1.

要创建 IPVLAN 设备，请输入以下命令：

```
ip link add link real_NIC_device name IPVLAN_device type ipvlan mode l2
```

请注意：网络接口控制器（NIC）是将计算机连接到网络的一个硬件组件。

#### 例 13.1. 创建 IPVLAN 设备

```
ip link add link enp0s31f6 name my_ipvlan type ipvlan mode l2
ip link
47: my_ipvlan@enp0s31f6: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000 link/ether e8:6a:6e:8a:a2:44 brd ff:ff:ff:ff:ff:ff
```

2.

要给接口分配 IPv4 或 IPv6 地址，请输入以下命令：

```
ip addr add dev IPVLAN_device IP_address/subnet_mask_prefix
```

3.

如果在 L3 模式或 L3S 模式中配置 IPVLAN 设备, 请进行以下设置:

a.

在远程主机上为远程 peer 配置邻居设置:

```
ip neigh add dev peer_device IPVLAN_device_IP_address lladdr MAC_address
```

其中 *MAC\_address* 是 IPVLAN 设备所基于的实际网卡的 MAC 地址。

b.

使用以下命令为 L3 模式 配置 IPVLAN 设备:

```
ip route add dev <real_NIC_device> <peer_IP_address/32>
```

对于 L3S 模式:

```
ip route add dev real_NIC_device peer_IP_address/32
```

其中 *IP-address* 代表远程 peer 的地址。

4.

要设置活跃的 IPVLAN 设备, 请输入以下命令:

```
ip link set dev IPVLAN_device up
```

5.

要检查 IPVLAN 设备是否活跃, 请在远程主机中执行以下命令:

```
ping IP_address
```

其中 *IP\_address* 使用 IPVLAN 设备的 IP 地址。

## 第 14 章 配置 NETWORKMANAGER 以忽略某些设备

默认情况下，NetworkManager 管理除 /usr/lib/udev/rules.d/85-nm-unmanaged.rules 文件中描述的之外的所有设备。要忽略某些其他设备，您可以在 NetworkManager 中将它们配置为 unmanaged。

### 14.1. 永久将设备配置为网络管理器（NETWORKMANAGER）中非受管设备

您可以根据几个标准，将设备配置为 unmanaged，如接口名称、MAC 地址或设备类型。如果您使用配置中的 device 部分将设备设置为非受管设备，NetworkManager 不会管理该设备，直到您开始使用该设备。

#### 流程

1.

可选：显示要识别设备的设备列表或您要设置为 unmanaged 的设备的 MAC 地址：

```
ip link show
...
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state
UP mode DEFAULT group default qlen 1000
 link/ether 52:54:00:74:79:56 brd ff:ff:ff:ff:ff:ff
...
...
```

2.

在 /etc/NetworkManager/conf.d/ 目录中创建 2.2.conf 文件，例如 /etc/NetworkManager/conf.d/99-unmanaged-devices.conf。

3.

对于您要配置为 unmanaged 的每个设备，为该文件添加一个带有唯一名称的部分。



**重要**

部分名称必须以 device- 开头。



要将特定接口配置为 unmanaged，请添加：

```
[device-enp1s0-unmanaged]
match-device=interface-name:enp1s0
managed=0
```

- 要将具有特定 MAC 地址的设备配置为 **unmanaged**, 请添加 :

```
[device-mac525400747956-unmanaged]
match-device=mac:52:54:00:74:79:56
managed=0
```

- 要将特定类型的所有设备配置为 **unmanaged**, 请添加 :

```
[device-ethernet-unmanaged]
match-device=type:ethernet
managed=0
```

- 要将多个设备设置为非受管设备, 请使用分号在 **unmanaged-devices** 参数中分隔条目, 例如 :

```
[device-multiple-devices-unmanaged]
match-device=interface-name:enp1s0;interface-name:enp7s0
managed=0
```

另外, 您可以为此文件中的每个设备添加单独的部分, 或者在 **/etc/NetworkManager/conf.d/** 目录中创建额外的 **5.2.conf** 文件。

4.

重启主机系统 :

```
reboot
```

验证

●

显示设备列表 :

```
nmcli device status
DEVICE TYPE STATE CONNECTION
enp1s0 ethernet unmanaged --
...
...
```

enp1s0 设备旁边的 **unmanaged** 状态表示 NetworkManager 没有管理该设备。

故障排除

- 如果 `nmcli device status` 命令的输出没有将设备列为 **非受管**，则显示 NetworkManager 配置：

```
NetworkManager --print-config
...
[device-enp1s0-unmanaged]
match-device=interface-name:enp1s0
managed=0
...
```

如果输出与您配置的设置不匹配，请确保没有优先级较高的配置文件覆盖了您的设置。有关 NetworkManager 如何合并多个配置文件的详情，请查看系统中的 [NetworkManager.conf \(5\) 手册页](#)。

## 14.2. 将设备临时配置为在 NETWORKMANAGER 中不被管理

您可以临时将设备配置为 **unmanaged**，例如，出于测试目的。这个更改会保留重新加载并重启 NetworkManager `systemd` 服务，而不是系统重启。

### 流程

1.

可选：显示设备列表，以便识别您要将其设置为 **unmanaged** 的设备：

```
nmcli device status
DEVICE TYPE STATE CONNECTION
enp1s0 ethernet disconnected --
...
```

2.

将 `enp1s0` 设备设置为 **unmanaged** 状态：

```
nmcli device set enp1s0 managed no
```

### 验证

●

显示设备列表：

```
nmcli device status
DEVICE TYPE STATE CONNECTION
enp1s0 ethernet unmanaged --
...
```

**enp1s0** 设备旁边的 **unmanaged** 状态表示 NetworkManager 没有管理该设备。

## 其他资源

- 您系统上的 **NetworkManager.conf (5)** 手册页

## 第 15 章 使用 NMCLI 配置回环接口

默认情况下，NetworkManager 不管理回环(`lo`)接口。为 `lo` 接口创建连接配置文件后，您可以使用 NetworkManager 配置这个设备。一些示例如下：

- 为 `lo` 接口分配额外的 IP 地址
- 定义 DNS 地址
- 更改 `lo` 接口的最大传输单元(MTU)大小

### 流程

1. 创建一个新的类型 `loopback`：

```
nmcli connection add con-name example-loopback type loopback
```

2. 配置自定义连接设置，例如：

- a. 要为接口分配额外的 IP 地址，请输入：

```
nmcli connection modify example-loopback +ipv4.addresses 192.0.2.1/24
```



#### 注意

NetworkManager 通过始终分配重启后仍保留的 IP 地址 `127.0.0.1` 和 `::1` 来管理 `lo` 接口。您不能覆盖 `127.0.0.1` 和 `::1`。但是，您可以为接口分配额外的 IP 地址。

- b. 要设置自定义最大传输单元(MTU)，请输入：

```
nmcli con mod example-loopback loopback.mtu 16384
```

C.

要为您的 DNS 服务器设置 IP 地址, 请输入 :

```
nmcli connection modify example-loopback ipv4.dns 192.0.2.0
```

如果您在 **loopback** 连接配置文件中设置了一个 DNS 服务器, 则此条目总是在 **/etc/resolv.conf** 文件中。DNS 服务器条目保持独立, 无论主机是否在不同网络之间漫游。

3.

激活连接 :

```
nmcli connection up example-loopback
```

验证

1.

显示 lo 接口的设置 :

```
ip address show lo
```

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16384 qdisc noqueue state UNKNOWN group default qlen 1000
```

```
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00 inet 127.0.0.1/8 scope host lo
valid_lft forever preferred_lft forever inet 192.0.2.1/24 brd 192.0.2.255 scope global lo
valid_lft forever preferred_lft forever
```

```
inet6 ::1/128 scope host
valid_lft forever preferred_lft forever
```

2.

验证 DNS 地址 :

```
cat /etc/resolv.conf
```

```
...
```

```
nameserver 192.0.2.0
```

```
...
```

## 第 16 章 创建一个虚拟接口

作为 Red Hat Enterprise Linux 用户，您可以创建并使用虚拟网络接口进行调试和测试。虚拟接口提供一个设备来路由数据包而无需实际传送数据包。它可让您创建使用网络管理器（NetworkManager）管理的其他回送设备，使不活跃 SLIP（Serial Line Internet Protocol）地址类似本地程序的实际地址。

### 16.1. 使用 NMCLI 创建一个同时具有 IPV4 和 IPV6 地址的虚拟接口

您可以创建一个带有各种设置的虚拟接口，如 IPv4 和 IPv6 地址。创建接口后，NetworkManager 会自动将其分配给默认的公共 firewalld 区域。

#### 流程

- 创建一个具有静态 IPv4 和 IPv6 地址，名为 `dummy0` 的伪接口：

```
nmcli connection add type dummy ifname dummy0 ipv4.method manual
ipv4.addresses 192.0.2.1/24 ipv6.method manual ipv6.addresses 2001:db8:2::1/64
```



#### 注意

要配置没有 IPv4 和 IPv6 地址的伪接口，请将 `ipv4.method` 和 `ipv6.method` 参数设置为 `disabled`。否则，IP 自动配置将失败，NetworkManager 会停用连接，并删除设备。

#### 验证

- 列出连接配置文件：

```
nmcli connection show
NAME UUID TYPE DEVICE
dummy-dummy0 aaf6eb56-73e5-4746-9037-eed42caa8a65 dummy dummy0
```

#### 其他资源

- 您系统上的 `nm-settings (5)` 手册页

## 第 17 章 使用 NETWORKMANAGER 为特定连接禁用 IPV6

在使用 NetworkManager 来管理网络接口的系统上，如果网络只使用 IPv4，您可以禁用 IPv6 协议。如果您禁用了 IPv6，NetworkManager 会自动在内核中设置相应的 sysctl 值。



### 注意

如果使用内核可调参数或内核引导参数禁用 IPv6，则必须额外考虑系统配置。如需更多信息，请参阅红帽知识库解决方案 [如何在 RHEL 中禁用或启用 IPv6 协议](#)。

### 17.1. 使用 NMCLI 禁用连接上的 IPV6

您可以使用 nmcli 工具在命令行上禁用 IPv6 协议。

#### 前提条件

- 系统使用 NetworkManager 来管理网络接口。

#### 流程

- 可选：显示网络连接的列表：

```
nmcli connection show
NAME UUID TYPE DEVICE
Example 7a7e0151-9c18-4e6f-89ee-65bb2d64d365 ethernet enp1s0
...
```

- 将连接的 ipv6.method 参数设为 disabled：

```
nmcli connection modify Example ipv6.method "disabled"
```

- 重启网络连接：

```
nmcli connection up Example
```

#### 验证

1.

显示设备的 IP 设置：

```
ip address show enp1s0
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state
UP group default qlen 1000
 link/ether 52:54:00:6b:74:be brd ff:ff:ff:ff:ff:ff
 inet 192.0.2.1/24 brd 192.10.2.255 scope global noprefixroute enp1s0
 valid_lft forever preferred_lft forever
```

如果没有显示 `inet6` 条目，则 IPv6 在该设备上被禁用。

2.

验证 `/proc/sys/net/ipv6/conf/enp1s0/disable_ipv6` 文件现在是否包含值 1：

```
cat /proc/sys/net/ipv6/conf/enp1s0/disable_ipv6
1
```

值 1 表示针对该设备禁用 IPv6。

## 第 18 章 更改主机名

系统的主机名是系统本身的名称。您可在安装 RHEL 时设置名称，之后可以更改它。

### 18.1. 使用 NMCLI 更改主机名

您可以使用 nmcli 工具更新系统主机名。请注意，其他工具可能会使用不同的术语，如静态或持久主机名。

#### 流程

1.

可选：显示当前主机名设置：

```
nmcli general hostname
old-hostname.example.com
```

2.

设置新主机名：

```
nmcli general hostname new-hostname.example.com
```

3.

NetworkManager 自动重启 `systemd-hostnamed` 来激活新主机名。要使更改生效，请重启主机：

```
reboot
```

或者，如果您知道哪个服务使用主机名：

a.

重启在服务启动时仅读取主机名的所有服务：

```
systemctl restart <service_name>
```

b.

活跃的 shell 用户必须重新登录才能使更改生效。

#### 验证

- 

显示主机名：

```
nmcli general hostname
new-hostname.example.com
```

## 18.2. 使用 HOSTNAMECTL 更改主机名

您可以使用 `hostnamectl` 工具更新主机名。默认情况下，这个工具设置以下主机名类型：

- 

静态主机名：存储在 `/etc/hostname` 文件中。通常，服务使用此名称作为主机名。

- 

用户友善的主机名：一个描述性名称，如 数据中心 A 中的代理服务器。

- 

临时主机名：通常从网络配置接收的回退值。

### 流程

1.

可选：显示当前主机名设置：

```
hostnamectl status --static
old-hostname.example.com
```

2.

设置新主机名：

```
hostnamectl set-hostname new-hostname.example.com
```

这个命令将静态和临时主机名设置为新值。要只设置一个特定类型，请将 `--static`、`--pretty` 或 `--transient` 选项传给命令。

3.

`hostnamectl` 工具自动重启 `systemd-hostnamed` 来激活新主机名。要使更改生效，请重启主机：

```
reboot
```

或者，如果您知道哪个服务使用主机名：

a.

重启在服务启动时仅读取主机名的所有服务：

```
systemctl restart <service_name>
```

b.

活跃的 shell 用户必须重新登录才能使更改生效。

## 验证



显示主机名：

```
hostnamectl status --static
new-hostname.example.com
```

## 第 19 章 配置 NETWORKMANAGER DHCP 设置

NetworkManager 提供与 DHCP 相关的不同的配置选项。例如，您可以将 NetworkManager 配置为使用内置的 DHCP 客户端（默认）或外部客户端，您可以影响单个配置文件的 DHCP 设置。

### 19.1. 更改 NETWORKMANAGER 的 DHCP 客户端

默认情况下，NetworkManager 使用其内部的 DHCP 客户端。但是，如果您需要不提供内置客户端的 DHCP 客户端，您也可以将 NetworkManager 配置为使用 dhclient。

请注意，RHEL 不提供 dhpcd，因此 NetworkManager 无法使用这个客户端。

#### 流程

1.

使用以下内容创建 /etc/NetworkManager/conf.d/dhcp-client.conf 文件：

```
[main]
dhcp=dhclient
```

您可以对 internal（默认）或 dhclient 设置 dhcp 参数。

2.

如果对 dhclient 设置 dhcp 参数，请安装 dhcp-client 软件包：

```
yum install dhcp-client
```

3.

重启 NetworkManager：

```
systemctl restart NetworkManager
```

请注意，重启会临时中断所有网络连接。

#### 验证

- 

在 `/var/log/messages` 日志文件中搜索类似于如下的条目：

```
Apr 26 09:54:19 server NetworkManager[27748]: <info> [1650959659.8483] dhcpc-init:
Using DHCP client 'dhclient'
```

此日志条目确认 NetworkManager 使用 `dhclient` 作为 DHCP 客户端。

## 其他资源

- 

您系统上的 `NetworkManager.conf (5)` 手册页

## 19.2. 配置 NETWORKMANAGER 连接的 DHCP 超时行为

DHCP 客户端在每次连接到网络时都从 DHCP 服务器请求动态 IP 地址和对应配置信息。

当您在连接配置文件中启用 DHCP 时，NetworkManager 默认等待 45 秒，以便此请求完成。

## 先决条件

- 

在主机上配置了使用 DHCP 的连接。

## 流程

1.

可选：设置 `ipv4.dhcp-timeout` 和 `ipv6.dhcp-timeout` 属性。例如，要将这两个选项都设为 30 秒，请输入：

```
nmcli connection modify <connection_name> ipv4.dhcp-timeout 30 ipv6.dhcp-
timeout 30
```

另外，将参数设置为 `infinity` 以配置网络管理器(NetworkManager)不会停止尝试请求和续订 IP 地址，直到成功为止。

2.

可选：配置如果网络管理器（NetworkManager）在超时前没有接收 IPv4 地址时的行为：

```
nmcli connection modify <connection_name> ipv4.may-fail <value>
```

如果将 `ipv4.may-fail` 选项设为：

- - `yes`, 连接的状态取决于 IPv6 配置：
    - 如果启用了 IPv6 配置并成功, NetworkManager 会激活 IPv6 连接, 不再尝试激活 IPv4 连接。
    - 如果禁用或未配置 IPv6 配置, 连接会失败。
  - `no`, 连接会被停止。在这种情况下：
    - 如果启用了连接的 `autoconnect` 属性, NetworkManager 会根据 `autoconnect-retries` 属性中设置的值尝试多次激活连接。默认值为 4。
    - 如果连接仍然无法获得 DHCP 地址, 则自动激活会失败。请注意, 5 分钟后, 自动连接过程会再次启动, 从 DHCP 服务器获取 IP 地址。
- 3. 可选：配置如果网络管理器（NetworkManager）在超时前没有接收 IPv6 地址时的行为：

```
nmcli connection modify <connection_name> ipv6.may-fail <value>
```

## 其他资源

- 您系统上的 `nm-settings` (5) 手册页

## 第 20 章 使用 NETWORKMANAGER 的一个分配程序脚本运行 HCLIENT EXIT HOOKS

您可以使用 NetworkManager 分配程序脚本来执行 dhclient exit hooks。

### 20.1. NETWORKMANAGER 分配程序脚本的概念

在发生网络事件时，NetworkManager-dispatcher 服务会按字母顺序执行用户提供的脚本。这些脚本通常是 shell 脚本，但可以是任何可执行的脚本或应用程序。例如，您可以使用分配程序脚本来调整您无法使用 NetworkManager 进行管理的与网络相关的设置。

您可以在以下目录中存储分配程序脚本：

- /etc/NetworkManager/dispatcher.d/ : root 用户可以编辑的分配程序脚本的通用位置。
- /usr/lib/NetworkManager/dispatcher.d/: 用于预先部署的不可变分配程序脚本。

为了安全起见，NetworkManager-dispatcher 服务只有在满足以下条件时才执行脚本：

- 脚本归 root 用户所有。
- 该脚本仅可由 root 读写。
- 脚本上没有设置 setuid 位。

NetworkManager-dispatcher 服务使用两个参数运行每个脚本：

1. 操作所在的设备的接口名称。
2. 当接口被激活时，如 up 操作。

NetworkManager(8) 手册页中的 分配程序脚本 部分提供了在脚本中可以使用的操作和环境变量的概述。

NetworkManager-dispatcher 服务一次运行一个脚本，但与主 NetworkManager 进程异步运行。请注意，如果脚本已排队，服务将始终运行它，即使后续事件使其过时。但是，NetworkManager-dispatcher 服务运行脚本，它们是指向 /etc/NetworkManager/dispatcher.d/no-wait.d/ 中的文件的符号链接，而无需等待之前脚本的终止，且并行运行。

## 其他资源

- 您系统上的 NetworkManager (8) 手册页

## 20.2. 创建运行 DHCLIENT EXIT HOOKS 的 NETWORKMANAGER 分配程序脚本

当 DHCP 服务器分配或更新 IPv4 地址时，NetworkManager 可以运行存储在 /etc/dhcp/dhclient-exit-hooks.d/ 目录中的分配程序脚本。然后，该分配程序脚本可以运行 dhclient 退出钩子。

### 先决条件

- dhclient exit hooks 存储在 /etc/dhcp/dhclient-exit-hooks.d/ 目录下。

### 流程

1. 使用以下内容创建 /etc/NetworkManager/dispatcher.d/12-dhclient-down 文件：

```
#!/bin/bash
Run dhclient.exit-hooks.d scripts

if [-n "$DHCP4_DHCPLEASETIME"] ; then
 if ["$2" = "dhcp4-change"] || ["$2" = "up"] ; then
 if [-d /etc/dhcp/dhclient-exit-hooks.d] ; then
 for f in /etc/dhcp/dhclient-exit-hooks.d/*.sh ; do
 if [-x "${f}"] ; then
 . "${f}"
 fi
 done
 fi
 fi
fi
```

2.

将 root 用户设为文件的所有者：

```
chown root:root /etc/NetworkManager/dispatcher.d/12-dhclient-down
```

3.

设置权限，以便只有 root 用户才能执行它：

```
chmod 0700 /etc/NetworkManager/dispatcher.d/12-dhclient-down
```

4.

恢复 SELinux 上下文：

```
restorecon /etc/NetworkManager/dispatcher.d/12-dhclient-down
```

## 其他资源



您系统上的 NetworkManager (8) 手册页

## 第 21 章 手动配置 /ETC/RESOLV.CONF 文件

默认情况下，NetworkManager 使用活跃的 NetworkManager 连接配置文件中的 DNS 设置动态更新 /etc/resolv.conf 文件。但是，您可以在 /etc/resolv.conf 中禁用此行为，并手动配置 DNS 设置。



### 注意

或者，如果您在 /etc/resolv.conf 中需要特定的 DNS 服务器顺序，请参阅 [配置 DNS 服务器的顺序](#)。

### 21.1. 在 NETWORKMANAGER 配置中禁用 DNS 处理

默认情况下，NetworkManager 在 /etc/resolv.conf 文件中管理 DNS 设置，您可以配置 DNS 服务器的顺序。或者，如果您喜欢在 /etc/resolv.conf 中手动配置 DNS 设置，您可以在 NetworkManager 中禁用 DNS 处理。

### 流程

1.

以 root 用户身份，使用文本编辑器创建包含以下内容的 /etc/NetworkManager/conf.d/90-dns-none.conf 文件：

```
[main]
dns=none
```

2.

重新载入 NetworkManager 服务：

```
systemctl reload NetworkManager
```



### 注意

重新加载服务后，NetworkManager 不再更新 /etc/resolv.conf 文件。但是该文件的最后内容将被保留。

3.

可选：从 /etc/resolv.conf 中删除 NetworkManager 生成的注释以避免混淆。

### 验证

1. 编辑 `/etc/resolv.conf` 文件并手动更新配置。
2. 重新载入 NetworkManager 服务：

```
systemctl reload NetworkManager
```

3. 显示 `/etc/resolv.conf` 文件：

```
cat /etc/resolv.conf
```

如果您成功禁用了 DNS 处理，NetworkManager 不会覆盖手动配置的设置。

## 故障排除

- 显示 NetworkManager 配置，以确保没有具有高优先级的配置文件覆盖设置：

```
NetworkManager --print-config
...
dns=none
...
```

## 其他资源

- 您系统上的 `NetworkManager.conf (5)` 手册页
- [使用 NetworkManager 配置 DNS 服务器的顺序](#)

### 21.2. 使用符号链接替换 /ETC/RESOLV.CONF 来手动配置 DNS 设置

默认情况下，NetworkManager 在 `/etc/resolv.conf` 文件中管理 DNS 设置，您可以配置 DNS 服务器的顺序。或者，如果您喜欢在 `/etc/resolv.conf` 中手动配置 DNS 设置，您可以在 NetworkManager 中禁用 DNS 处理。例如，如果 `/etc/resolv.conf` 是一个符号链接，则 NetworkManager 不会自动更新 DNS 配置。

## 前提条件

- NetworkManager rc-manager 配置选项没有设置为 file。要验证，请使用 NetworkManager --print-config 命令。

## 流程

1. 创建一个文件，如 /etc/resolv.conf.manually-configured，并将您环境的 DNS 配置添加到其中。使用与原始 /etc/resolv.conf 中一样的参数和语法。
2. 删除 /etc/resolv.conf 文件：

```
rm /etc/resolv.conf
```

3. 创建名为 /etc/resolv.conf 的符号链接，该链接指向 /etc/resolv.conf.manually-configured  
：

```
ln -s /etc/resolv.conf.manually-configured /etc/resolv.conf
```

## 其他资源

- 您系统上的 resolv.conf (5) 和 NetworkManager.conf (5) 手册页
- 使用 NetworkManager 配置 DNS 服务器的顺序

## 第 22 章 配置 DNS 服务器顺序

大多数应用程序使用 glibc 库的 `getaddrinfo()` 函数来解析 DNS 的名称。默认情况下，glibc 将所有 DNS 请求发送到 `/etc/resolv.conf` 文件中指定的第一个 DNS 服务器。如果这个服务器没有回复，RHEL 会尝试此文件中的所有其他名称服务器。NetworkManager 可让您影响 `/etc/resolv.conf` 中的 DNS 服务器顺序。

### 22.1. NETWORKMANAGER 如何在 /ETC/RESOLV.CONF 中对 DNS 服务器进行排序

NetworkManager 根据以下规则对 `/etc/resolv.conf` 文件中的 DNS 服务器进行排序：

- 如果只有一个连接配置集，NetworkManager 将使用那个连接中指定的 IPv4 和 IPv6 DNS 服务器顺序。
- 如果激活多个连接配置集，NetworkManager 会根据 DNS 优先级值对 DNS 服务器进行排序。如果您设置了 DNS 优先级，NetworkManager 的行为取决于 `dns` 参数中设置的值。您可以在 `/etc/NetworkManager/NetworkManager.conf` 文件的 `[main]` 部分中设置此参数：
  - `dns=default` 或者如果 `dns` 参数没有设置：  
NetworkManager 根据每个连接中的 `ipv4.dns-priority` 和 `ipv6.dns-priority` 参数对不同连接中的 DNS 服务器进行排序。  
  
如果没有设置值，或者您将 `ipv4.dns-priority` 和 `ipv6.dns-priority` 设为 0，则 NetworkManager 将使用全局默认值。请参阅 [DNS 优先级参数的默认值](#)。
  - `dns=dnsmasq` 或 `dns=systemd-resolved`：  
当您使用这些设置中的一个时，NetworkManager 将 `dnsmasq` 的 127.0.0.1 或 127.0.0.53 设为 `/etc/resolv.conf` 文件中的 `nameserver` 条目。  
  
`dnsmasq` 和 `systemd-resolved` 服务将对 NetworkManager 连接中设置的搜索域的查询转发到该连接中指定的 DNS 服务器，并将对其他域的查询转发到具有默认路由的连接。当

多个连接有相同的搜索域集时，`dnsmasq` 和 `systemd-resolved` 将这个域的查询转发到在具有最低优先级值的连接中设置的 DNS 服务器。

## DNS 优先级参数的默认值

`NetworkManager` 对连接使用以下默认值：

- 50 用于 VPN 连接
- 100 用于其他连接

## 有效的 DNS 优先级值：

您可以将全局默认值和特定于连接的 `ipv4.dns-priority` 和 `ipv6.dns-priority` 参数设为 -2147483647 和 2147483647 之间的值。

- 低的值具有更高的优先级。
- 负值具有一个特殊的效果，它会排除其他带有更大值的配置。例如，如果至少有一个连接具有负优先级值，`NetworkManager` 只使用在连接配置集中指定的具有最低优先级的 DNS 服务器。
- 如果多个连接具有相同的 DNS 优先级，`NetworkManager` 会按照以下顺序排列 DNS 的优先顺序：
  - a. **VPN** 连接
  - b. 带有活跃的默认路由的连接。活跃的默认路由是具有最低指标的默认路由。

## 其他资源

- 您系统上的 **nm-settings (5)** 手册页
- [在不同域中使用不同的 DNS 服务器](#)

## 22.2. 设置 NETWORKMANAGER 范围默认 DNS 服务器优先级值

NetworkManager 为连接使用以下 DNS 优先级默认值：

- 50 用于 VPN 连接
- 100 用于其他连接

您可以对 IPv4 和 IPv6 连接使用自定义的默认值覆盖这些系统范围的默认值。

### 流程

1.

编辑 /etc/NetworkManager/NetworkManager.conf 文件：

a.

添加 [connection] 部分（如果其不存在）：

```
[connection]
```

b.

将自定义默认值添加到 [connection] 部分。例如，要将 IPv4 和 IPv6 的新默认值设为 200，请添加：

```
ipv4.dns-priority=200
ipv6.dns-priority=200
```

您可以将参数设为 -2147483647 和 2147483647 之间的值。请注意，将参数设置为 0 将启用内置的默认值（对于 VPN 连接为 50，对于其他连接为 100）。

2.

重新载入 NetworkManager 服务：

```
systemctl reload NetworkManager
```

### 其他资源

- 您系统上的 NetworkManager.conf (5) 手册页

### 22.3. 设置网络管理器连接的 DNS 优先级

如果您需要 DNS 服务器的特定顺序，您可以在连接配置文件中设置优先级值。NetworkManager 使用这些值来在服务创建或更新 /etc/resolv.conf 文件时对服务器进行排序。

请注意，只有在您配置了多个与不同 DNS 服务器的连接时，设置 DNS 优先级才有意义。如果您只有一个与多个 DNS 服务器的连接，请在连接配置集中按首选顺序手动设置 DNS 服务器。

#### 先决条件

- 系统配置了多个网络管理器连接。
- 系统在 /etc/NetworkManager/NetworkManager.conf 文件中未设置 dns 参数，或者该参数被设为了 default。

#### 流程

1.

可选：显示可用的连接：

```
nmcli connection show
NAME UUID TYPE DEVICE
Example_con_1 d17ee488-4665-4de2-b28a-48befab0cd43 ethernet enp1s0
Example_con_2 916e4f67-7145-3ffa-9f7b-e7cada8f6bf7 ethernet enp7s0
...
```

2.

设置 ipv4.dns-priority 和 ipv6.dns-priority 参数。例如，要将这两个参数都设置为 10，请输入：

```
nmcli connection modify <connection_name> ipv4.dns-priority 10 ipv6.dns-priority
10
```

3.

可选：对其他连接重复上一步。

4.

重新激活您更新的连接：

```
nmcli connection up <connection_name>
```

验证

•

显示 /etc/resolv.conf 文件的内容以验证 DNS 服务器的顺序是否正确：

```
cat /etc/resolv.conf
```

## 第 23 章 在不同域中使用不同的 DNS 服务器

默认情况下，Red Hat Enterprise Linux (RHEL) 将所有 DNS 请求发送到 /etc/resolv.conf 文件中指定的第一个 DNS 服务器。如果这个服务器没有回复，RHEL 会尝试此文件中的下一个服务器，直到找到一个正常工作的为止。在一个 DNS 服务器无法解析所有域的环境中，管理员可将 RHEL 配置为将特定域的 DNS 请求发送到所选 DNS 服务器。

例如，您要将服务器连接到虚拟专用网络(VPN)，VPN 中的主机使用 example.com 域。在这种情况下，您可以以下方式配置 RHEL，以处理 DNS 查询：

- 仅将 example.com 的 DNS 请求发送到 VPN 网络中的 DNS 服务器。
- 将所有其他请求发送到使用默认网关在连接配置文件中配置的 DNS 服务器。

### 23.1. 在 NETWORKMANAGER 中使用 DNMASQ 将特定域的 DNS 请求发送到所选的 DNS 服务器

您可以将 NetworkManager 配置为启动 dnsmasq 的实例。然后，此 DNS 缓存服务器侦听 loopback 设备上的端口 53。因此，该服务只能从本地系统访问，而不可从网络访问。

使用这个配置，NetworkManager 将 nameserver 127.0.0.1 条目添加到 /etc/resolv.conf 文件中，dnsmasq 会动态将 DNS 请求路由 NetworkManager 连接配置文件中指定的相应的 DNS 服务器。

#### 先决条件

- 系统配置了多个网络管理器连接。
- 在负责解析特定域的 NetworkManager 连接配置文件中配置了 DNS 服务器和搜索域。

例如，要确保 VPN 连接中指定的 DNS 服务器解析对 example.com 域的查询，VPN 连接配置文件必须包含以下设置：

- 可以在 ipv4.dns 和 ipv6.dns 参数中解析 example.com 的 DNS 服务器

- 在 `ipv4.dns-search` 和 `ipv6.dns-search` 参数中设置为 `example.com` 的搜索域
- `dnsmasq` 服务没有运行，或配置为在与 `localhost` 不同的接口上侦听。

## 流程

1. 安装 `dnsmasq` 软件包：

```
yum install dnsmasq
```
2. 编辑 `/etc/NetworkManager/NetworkManager.conf` 文件，并在 `[main]` 部分中设置以下条目：

```
dns=dnsmasq
```
3. 重新载入 `NetworkManager` 服务：

```
systemctl reload NetworkManager
```

## 验证

1. 在 `NetworkManager` 单元的 `systemd` 日志中搜索服务使用不同的 DNS 服务器的域：

```
journalctl -xeu NetworkManager
...
Jun 02 13:30:17 <client_hostname>_dnsmasq[5298]: using nameserver
198.51.100.7#53 for domain example.com
...
```
2. 使用 `tcpdump` 数据包嗅探器来验证 DNS 请求的正确路由：
  - a. 安装 `tcpdump` 软件包：

```
yum install tcpdump
```

b.

在一个终端上，启动 **tcpdump** 以捕获所有接口上的 DNS 流量：

```
tcpdump -i any port 53
```

c.

在另一个终端上，为存在异常的域解析主机名，为另一个域解析主机名，例如：

```
host -t A www.example.com
host -t A www.redhat.com
```

d.

在 **tcpdump** 输出中验证 Red Hat Enterprise Linux 是否只向指定的 DNS 服务器并通过相应的接口发送对 example.com 域的 DNS 查询：

```
...
13:52:42.234533 IP server.43534 > 198.51.100.7.domain: 50121+ [1au] A?
www.example.com. (33)
...
13:52:57.753235 IP server.40864 > 192.0.2.1.domain: 6906+ A? www.redhat.com.
(33)
...
```

Red Hat Enterprise Linux 将对 www.example.com 的 DNS 查询发送到 198.51.100.7 上的 DNS 服务器，将对 www.redhat.com 的查询发送到 192.0.2.1。

## 故障排除

1.

验证 /etc/resolv.conf 文件中的 nameserver 条目是否指向 127.0.0.1：

```
cat /etc/resolv.conf
nameserver 127.0.0.1
```

如果缺少条目，请检查 /etc/NetworkManager/NetworkManager.conf 文件中的 dns 参数。

2.

验证 dnsmasq 服务是否侦听 loopback 设备上的端口 53：

```
ss -tulpn | grep "127.0.0.1:53"
udp UNCONN 0 0 127.0.0.1:53 0.0.0.0:* users:(("dnsmasq",pid=7340,fd=18))
tcp LISTEN 0 32 127.0.0.1:53 0.0.0.0:* users:(("dnsmasq",pid=7340,fd=19))
```

如果服务没有侦听 127.0.0.1:53， 请检查 NetworkManager 单元的日志条目：

```
journalctl -u NetworkManager
```

## 23.2. 使用 NETWORKMANAGER 中的 SYSTEMD-RESOLVED 将对特定域的 DNS 请求发送到所选的 DNS 服务器

您可以将 NetworkManager 配置为启动 `systemd-resolved` 的实例。然后，这个 DNS stub 解析器监听 IP 地址 127.0.0.53 上的端口 53。因此，这个 stub 解析器只能从本地系统访问，而不可从网络访问。

使用这个配置，NetworkManager 将 `nameserver 127.0.0.53` 条目添加到 `/etc/resolv.conf` 文件中，`systemd-resolved` 将 DNS 请求动态路由到 NetworkManager 连接配置文件中指定的相应的 DNS 服务器。



### 重要

`systemd-resolved` 服务仅作为技术预览提供。红帽产品服务级别协议（SLA）不支持技术预览功能，且其功能可能并不完善，因此红帽不建议在生产环境中使用它们。这些预览可让用户早期访问将来的产品功能，让用户在开发过程中测试并提供反馈意见。

如需有关 [技术预览功能支持范围](#) 的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

对于支持的解决方案，请参阅 [在 NetworkManager 中使用 dnsmasq 将对特定域的 DNS 请求发送到所选的 DNS 服务器](#)。

### 先决条件

- 系统配置了多个网络管理器连接。
- 在负责解析特定域的 NetworkManager 连接配置文件中配置了 DNS 服务器和搜索域。

例如，要确保 VPN 连接中指定的 DNS 服务器解析对 `example.com` 域的查询，VPN 连接配置文件必须包含以下设置：

- 可以在 `ipv4.dns` 和 `ipv6.dns` 参数中解析 `example.com` 的 DNS 服务器
- 在 `ipv4.dns-search` 和 `ipv6.dns-search` 参数中设置为 `example.com` 的搜索域

## 流程

1. 安装 `systemd-resolved` 软件包：

```
dnf install systemd-resolved
```

2. 启用并启动 `systemd-resolved` 服务：

```
systemctl --now enable systemd-resolved
```

3. 编辑 `/etc/NetworkManager/NetworkManager.conf` 文件，并在 `[main]` 部分中设置以下条目：

```
dns=systemd-resolved
```

4. 重新载入 `NetworkManager` 服务：

```
systemctl reload NetworkManager
```

## 验证

1. 显示 `systemd-resolved` 使用的 DNS 服务器，以及服务对哪个域使用不同的 DNS 服务器：

```
resolvectl
...
Link 2 (enp1s0)
 Current Scopes: DNS
 Protocols: +DefaultRoute ...
 Current DNS Server: 192.0.2.1
 DNS Servers: 192.0.2.1

Link 3 (tun0)
 Current Scopes: DNS
 Protocols: -DefaultRoute ...
```

```
Current DNS Server: 198.51.100.7
DNS Servers: 198.51.100.7 203.0.113.19
DNS Domain: example.com
```

输出确认 `systemd-resolved` 为 `example.com` 域使用不同的 DNS 服务器。

2.

使用 `tcpdump` 数据包嗅探器来验证 DNS 请求的正确路由：

a.

安装 `tcpdump` 软件包：

```
yum install tcpdump
```

b.

在一个终端上，启动 `tcpdump` 以捕获所有接口上的 DNS 流量：

```
tcpdump -i any port 53
```

c.

在另一个终端上，为存在异常的域解析主机名，为另一个域解析主机名，例如：

```
host -t A www.example.com
host -t A www.redhat.com
```

d.

在 `tcpdump` 输出中验证 Red Hat Enterprise Linux 是否只向指定的 DNS 服务器并通过相应的接口发送对 `example.com` 域的 DNS 查询：

```
...
13:52:42.234533 IP server.43534 > 198.51.100.7.domain: 50121+ [1au] A?
www.example.com. (33)
...
13:52:57.753235 IP server.40864 > 192.0.2.1.domain: 6906+ A? www.redhat.com.
(33)
...
```

Red Hat Enterprise Linux 将对 `www.example.com` 的 DNS 查询发送到 198.51.100.7 上的 DNS 服务器，将对 `www.redhat.com` 的查询发送到 192.0.2.1。

1.

验证 `/etc/resolv.conf` 文件中的 `nameserver` 条目是否指向 127.0.0.53：

```
cat /etc/resolv.conf
nameserver 127.0.0.53
```

如果缺少条目，请检查 `/etc/NetworkManager/NetworkManager.conf` 文件中的 `dns` 参数。

2.

验证 `systemd-resolved` 服务是否监听本地 IP 地址 127.0.0.53 上的端口 53：

```
ss -tulpn | grep "127.0.0.53"
udp UNCONN 0 0 127.0.0.53%lo:53 0.0.0.0:* users:(("systemd-
resolve",pid=1050,fd=12))
tcp LISTEN 0 4096 127.0.0.53%lo:53 0.0.0.0:* users:(("systemd-
resolve",pid=1050,fd=13))
```

如果服务没有侦听 127.0.0.53:53，请检查 `systemd-resolved` 服务是否在运行。

## 第 24 章 管理默认网关设置

默认网关是在没有其他路由与数据包的目的地匹配时转发网络数据包的路由器。在本地网络中，默认网关通常是与距离互联网有一跳的主机。

### 24.1. 使用 NMCLI 对现有连接设置默认网关

在大多数情况下，管理员在创建连接时设置默认网关。但是，您也可以使用 nmcli 工具对之前创建的连接设置或更新默认网关设置。

#### 先决条件

- 至少需要在设置默认网关的连接上配置一个静态 IP 地址。
- 如果用户在物理控制台中登录，用户权限就足够了。否则，该用户必须具有 root 权限。

#### 流程

1. 设置默认网关的 IP 地址：

要设置 IPv4 默认网关，请输入：

```
nmcli connection modify <connection_name> ipv4.gateway
"<IPv4_gateway_address>"
```

要设置 IPv6 默认网关，请输入：

```
nmcli connection modify <connection_name> ipv6.gateway
"<IPv6_gateway_address>"
```

2. 重启网络连接以使更改生效：

```
nmcli connection up <connection_name>
```



## 警告

所有目前使用这个网络连接的连接在重启过程中暂时中断。

## 验证

- 验证路由是否处于活跃状态：

a.

要显示 IPv4 默认网关，请输入：

```
ip -4 route
default via 192.0.2.1 dev example proto static metric 100
```

b.

要显示 IPv6 默认网关，请输入：

```
ip -6 route
default via 2001:db8:1::1 dev example proto static metric 100 pref medium
```

## 24.2. 使用 NMCLI 交互模式对现有连接设置默认网关

在大多数情况下，管理员在创建连接时设置默认网关。但是，您也可以使用 nmcli 工具的交互模式对之前创建的连接设置或更新默认网关设置。

## 先决条件

- 至少需要在设置默认网关的连接上配置一个静态 IP 地址。
- 如果用户在物理控制台中登录，用户权限就足够了。否则，该用户必须具有 root 权限。

## 流程

- 为所需连接打开 nmcli 交互模式：

```
nmcli connection edit <connection_name>
```

2.

设置默认网关

要设置 IPv4 默认网关，请输入：

```
nmcli> set ipv4.gateway "<IPv4_gateway_address>"
```

要设置 IPv6 默认网关，请输入：

```
nmcli> set ipv6.gateway "<IPv6_gateway_address>"
```

3.

可选：验证默认网关是否被正确设置：

```
nmcli> print
...
ipv4.gateway: <IPv4_gateway_address>
...
ipv6.gateway: <IPv6_gateway_address>
...
```

4.

保存配置：

```
nmcli> save persistent
```

5.

重启网络连接以使更改生效：

```
nmcli> activate <connection_name>
```



警告

所有目前使用这个网络连接的连接在重启过程中暂时中断。

6.

保留 nmcli 交互模式：

```
nmcli> quit
```

验证

•

验证路由是否处于活跃状态：

a.

要显示 IPv4 默认网关，请输入：

```
ip -4 route
default via 192.0.2.1 dev example proto static metric 100
```

b.

要显示 IPv6 默认网关，请输入：

```
ip -6 route
default via 2001:db8:1::1 dev example proto static metric 100 pref medium
```

### 24.3. 使用 NM-CONNECTION-EDITOR 在现有连接上设置默认网关

在大多数情况下，管理员在创建连接时设置默认网关。但是，您也可以使用 `nm-connection-editor` 应用程序对之前创建的连接设置或更新默认网关设置。

先决条件

•

至少需要在设置默认网关的连接上配置一个静态 IP 地址。

流程

1.

打开一个终端，输入 `nm-connection-editor`：

```
nm-connection-editor
```

2.

选择要修改的连接，并点击 **gear wheel** 图标编辑现有连接。

3.

设置 IPv4 默认网关。例如，要将连接上的默认网关的 IPv4 地址设为 192.0.2.1：

a.

打开 IPv4 Settings 选项卡。

b.

在网关地址所在的 IP 范围旁边的 gateway 字段中输入地址：

Addresses		
Address	Netmask	Gateway
192.0.2.123	24	192.0.2.1

4.

设置 IPv6 默认网关。例如，要将连接上默认网关的 IPv6 地址设为 2001:db8:1::1：

a.

打开 IPv6 选项卡。

b.

在网关地址所在的 IP 范围旁边的 gateway 字段中输入地址：

Addresses		
Address	Prefix	Gateway
2001:db8:1::5	64	2001:db8:1::1

5.

点击 确定。

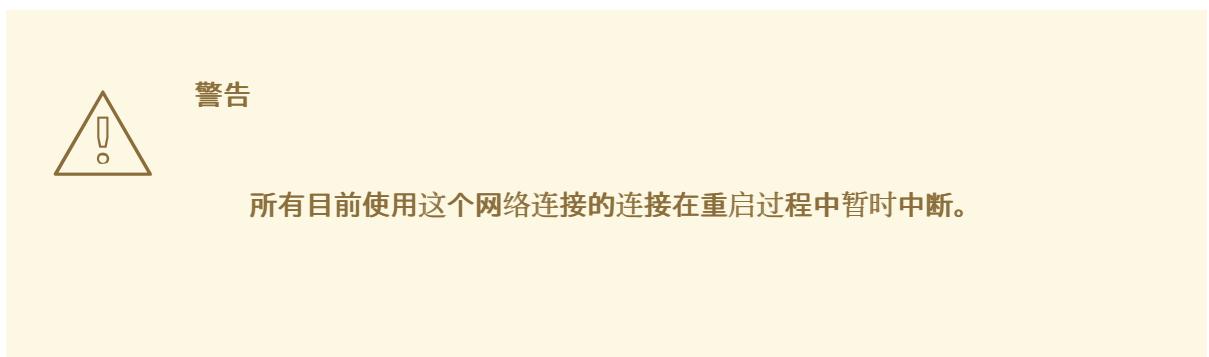
6.

点击 Save。

7.

重启网络连接以使更改生效。例如，要使用命令行重启 example 连接：

```
nmcli connection up example
```



## 验证

- 验证路由是否处于活跃状态。

显示 IPv4 默认网关：

```
ip -4 route
default via 192.0.2.1 dev example proto static metric 100
```

显示 IPv6 默认网关：

```
ip -6 route
default via 2001:db8:1::1 dev example proto static metric 100 pref medium
```

## 其他资源

- [使用 nm-connection-editor 配置以太网连接](#)

### 24.4. 使用 CONTROL-CENTER 在现有连接上设置默认网关

在大多数情况下，管理员在创建连接时设置默认网关。但是，您也可以使用 `control-center` 应用程序对之前创建的连接设置或更新默认网关设置。

## 先决条件

- 至少需要在设置默认网关的连接上配置一个静态 IP 地址。

- 连接的网络配置在 **control-center** 应用程序中打开。

## 流程

1. 设置 IPv4 默认网关。例如，要将连接上的默认网关的 IPv4 地址设为 192.0.2.1：

- a. 打开 IPv4 选项卡。
- b. 在网关地址所在的 IP 范围旁边的 gateway 字段中输入地址：

Addresses		
Address	Netmask	Gateway
192.0.2.123	255.255.255.0	192.0.2.1

2. 设置 IPv6 默认网关。例如，要将连接上默认网关的 IPv6 地址设为 2001:db8:1::1：

- a. 打开 IPv6 选项卡。
- b. 在网关地址所在的 IP 范围旁边的 gateway 字段中输入地址：

Addresses		
Address	Prefix	Gateway
2001:db8:1::5	64	2001:db8:1::1

3. 点应用。
4. 返回到 Network 窗口，通过将连接的按钮切换为 Off，然后返回到 On 来禁用并重新启用连

接，以使更改生效。

 警告

所有目前使用这个网络连接的连接在重启过程中暂时中断。

验证

- 验证路由是否处于活跃状态。

显示 IPv4 默认网关：

```
$ ip -4 route
default via 192.0.2.1 dev example proto static metric 100
```

显示 IPv6 默认网关：

```
$ ip -6 route
default via 2001:db8:1::1 dev example proto static metric 100 pref medium
```

其他资源

- [使用控制中心配置以太网连接](#)

## 24.5. 使用 NMSTATECTL 对现有连接设置默认网关

在大多数情况下，管理员在创建连接时设置默认网关。但是，您也可以使用 `nmstatectl` 工具对之前创建的连接设置或更新默认网关设置。

使用 `nmstatectl` 工具通过 Nmstate API 设置默认网关。Nmstate API 确保设置配置后结果与配置文件匹配。如果有任何失败，`nmstatectl` 会自动回滚更改以避免系统处于不正确的状态。

## 先决条件

- 至少需要在设置默认网关的连接上配置一个静态 IP 地址。
- `enp1s0` 接口已配置， 默认网关的 IP 地址在此接口的 IP 配置子网内。
- `nmstate` 软件包已安装。

## 流程

1.

创建一个包含以下内容的 YAML 文件，如 `~/set-default-gateway.yml`：

```

routes:
 config:
 - destination: 0.0.0.0/0
 next-hop-address: 192.0.2.1
 next-hop-interface: enp1s0
```

这些设置将 `192.0.2.1` 定义为默认网关，可通过 `enp1s0` 接口访问默认网关。

2.

将设置应用到系统：

```
nmstatectl apply ~/set-default-gateway.yml
```

## 其他资源

- 您系统上的 `nmstatectl (8)` 手册页
- `/usr/share/doc/nmstate/examples/` 目录

## 24.6. 使用 NETWORK RHEL 系统角色对现有连接设置默认网关

如果数据包的目的地无法通过直接连接的网络或通过主机上配置的任何路由到达，则主机将网络数据包转发到其默认网关。要配置主机的默认网关，请在连接到同一网络的接口的 NetworkManager 连接配置

文件中将其设置为默认网关。通过使用 Ansible 和 network RHEL 系统角色，您可以自动化此过程，并在 playbook 中定义的主机上远程配置连接配置文件。

在大多数情况下，管理员在创建连接时设置默认网关。但是，您也可以对之前创建的连接设置或更新默认网关设置。



### 警告

您不能使用 network RHEL 系统角色只更新现有连接配置文件中的特定值。该角色确保连接配置文件与 playbook 中的设置完全匹配。如果具有相同名称的连接配置文件存在，则角色将应用 playbook 中的设置，并将配置文件中的所有其他设置重置为其默认值。要防止重置值，请始终在 playbook 中指定网络连接配置文件的整个配置，包括您不想更改的设置。

## 先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 sudo 权限。

## 步骤

1. 创建一个包含以下内容的 playbook 文件，如 ~/playbook.yml：

```

- name: Configure the network
 hosts: managed-node-01.example.com
 tasks:
 - name: Ethernet connection profile with static IP address settings
 ansible.builtin.include_role:
 name: redhat.rhel_system_roles.network
 vars:
 network_connections:
 - name: enp1s0
```

```

type: ethernet
autoconnect: yes
ip:
 address:
 - 198.51.100.20/24
 - 2001:db8:1::1/64
 gateway4: 198.51.100.254
 gateway6: 2001:db8:1::fffe
 dns:
 - 198.51.100.200
 - 2001:db8:1::ffbb
 dns_search:
 - example.com
state: up

```

有关 playbook 中使用的所有变量的详情，请查看控制节点上的 /usr/share/ansible/roles/rhel-system-roles.network/README.md 文件。

2.

验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3.

运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

●

查询受管节点的 Ansible 事实，并验证活跃的网络设置：

```

ansible managed-node-01.example.com -m ansible.builtin.setup
...
 "ansible_default_ipv4": {
...
 "gateway": "198.51.100.254",
 "interface": "enp1s0",
...
 },
 "ansible_default_ipv6": {
...
 "gateway": "2001:db8:1::fffe",
 "interface": "enp1s0",

```

```
 ...
}
```

## 其他资源

- [/usr/share/ansible/roles/rhel-system-roles.network/README.md 文件](#)
- [/usr/share/doc/rhel-system-roles/network/ 目录](#)

## 24.7. 使用旧的网络脚本在现有连接中设置默认网关

在大多数情况下，管理员在创建连接时设置默认网关。但是，在使用旧的网络脚本时，您还可以在之前创建的连接上设置或更新默认网关设置。

### 前提条件

- NetworkManager 软件包未安装，或者 NetworkManager 服务被禁用。
- network-scripts 软件包已安装。

### 流程

1. 将 /etc/sysconfig/network-scripts/ifcfg-enp1s0 文件中的 GATEWAY 参数设为 192.0.2.1  
：  

```
GATEWAY=192.0.2.1
```
2. 在 /etc/sysconfig/network-scripts/route-enp0s1 文件中添加 default 条目：  

```
default via 192.0.2.1
```
3. 重启网络：  

```
systemctl restart network
```

## 24.8. NETWORKMANAGER 如何管理多个默认网关

在某些情况下，您可能需要在主机上设置多个默认网关。但是，为了避免异步路由问题，同一协议的每个默认网关都需要单独的指标值。请注意，RHEL 只使用到设置成最低指标的默认网关的连接。

您可以使用以下命令为连接的 IPv4 和 IPv6 网关设置指标：

```
nmcli connection modify <connection_name> ipv4.route-metric <value> ipv6.route-metric <value>
```



### 重要

不要为多个连接配置集中的同一协议设置相同的指标值以避免路由问题。

如果您设置了没有指标值的默认网关，则 NetworkManager 会自动根据接口类型设置指标值。为此，NetworkManager 将这个网络类型的默认值分配给激活的第一个连接，并根据激活的顺序为同一类型的每一个其他连接设置递增值。例如，如果带有默认网关的两个以太网连接存在，则 NetworkManager 会将路由上的 100 指标设置为您首先激活的连接的默认网关。对于第二个连接，NetworkManager 会设为 101。

以下是经常使用的网络类型及其默认指标的概述：

连接类型	默认指标值
VPN	50
Ethernet	100
MACsec	125
InfiniBand	150
Bond	300
Team	350
VLAN	400
Bridge	425

连接类型	默认指标值
TUN	450
Wi-Fi	600
IP tunnel	675

## 其他资源

- [配置基于策略的路由以定义其他路由](#)

## 24.9. 配置 NETWORKMANAGER 以避免使用特定配置集提供默认网关

您可以配置 NetworkManager 从不使用特定的配置文件来提供默认网关。对于没有连接到默认网关的连接配置集，请按照以下步骤操作。

### 先决条件

- 没有连接到默认网关的连接的 NetworkManager 连接配置文件存在。

### 流程

1. 如果连接使用动态 IP 配置，请配置 NetworkManager 不会使用这个连接作为相应 IP 类型的默认连接，这意味着 NetworkManager 永远不会为其分配默认路由：

```
nmcli connection modify <connection_name> ipv4.never-default yes ipv6.never-default yes
```

请注意，将 `ipv4.never-default` 和 `ipv6.never-default` 设为 `yes`，会自动从连接配置文件中删除相应协议默认网关的 IP 地址。

2. 激活连接：

```
nmcli connection up <connection_name>
```

### 验证

- 使用 ip -4 路由和 ip -6 route 命令，来验证 RHEL 是否未对 IPv4 和 IPv6 协议的默认路使用网络接口。

## 24.10. 修复因为多个默认网关导致的意外路由行为

只有一些场景，比如使用多路径 TCP 时，您需要在主机上有多个默认网关。在大多数情况下，您只配置一个默认网关，来避免意外的路由行为或异步路由问题。



### 注意

要将流量路由到不同的互联网提供商，请使用基于策略的路由，而不是多个默认网关。

### 先决条件

- 主机使用 NetworkManager 管理网络连接，这是默认设置。
- 主机有多个网络接口。
- 主机配置了多个默认网关。

### 流程

1.

显示路由表：

- 对于 IPv4，请输入：

```
ip -4 route
default via 192.0.2.1 dev enp1s0 proto dhcp metric 101
default via 198.51.100.1 dev enp7s0 proto dhcp metric 102
...
```

- 对于 IPv6，请输入：

```
ip -6 route
```

```
default via 2001:db8:1::1 dev enp1s0 proto static ra 101 pref medium
default via 2001:db8:2::1 dev enp7s0 proto static ra 102 pref medium
...
```

以 **default** 开头的条目表示默认路由。注意 **dev** 旁边显示的这些条目的接口名称。

2.

使用以下命令显示您使用在上一步中识别的接口的 NetworkManager 连接：

```
nmcli -f GENERAL.CONNECTION,IP4.GATEWAY,IP6.GATEWAY device show enp1s0
GENERAL.CONNECTION: Corporate-LAN
IP4.GATEWAY: 192.0.2.1
IP6.GATEWAY: 2001:db8:1::1

nmcli -f GENERAL.CONNECTION,IP4.GATEWAY,IP6.GATEWAY device show enp7s0
GENERAL.CONNECTION: Internet-Provider
IP4.GATEWAY: 198.51.100.1
IP6.GATEWAY: 2001:db8:2::1
```

在这些示例中，名为 **Corporate-LAN** 和 **Internet-Provider** 的配置文件设置了默认网关。因为在本地网络中，默认网关通常是距离互联网一跳的主机，所以此流程的剩下部分假设 **Corporate-LAN** 中的默认网关是不正确的。

3.

配置 NetworkManager 不使用 **Corporate-LAN** 连接作为 IPv4 和 IPv6 连接的默认路由：

```
nmcli connection modify Corporate-LAN ipv4.never-default yes ipv6.never-default
yes
```

请注意，将 **ipv4.never-default** 和 **ipv6.never-default** 设为 **yes**，会自动从连接配置文件中删除相应协议默认网关的 IP 地址。

4.

激活 **Corporate-LAN** 连接：

```
nmcli connection up Corporate-LAN
```

验证

- 

显示 IPv4 和 IPv6 路由表，并确认每个协议都只有一个默认网关：

- 

对于 IPv4, 请输入 :

```
ip -4 route
default via 198.51.100.1 dev enp7s0 proto dhcp metric 102
...
```

- 

对于 IPv6, 请输入 :

```
ip -6 route
default via 2001:db8:2::1 dev enp7s0 proto ra metric 102 pref medium
...
```

## 其他资源

- 

[配置基于策略的路由以定义其他路由](#)

## 第 25 章 配置静态路由

路由可确保您可以在相互连接的网络间发送和接收流量。在较大环境中，管理员通常配置服务以便路由器可以动态地了解其他路由器。在较小的环境中，管理员通常会配置静态路由，以确保流量可以从一个网络到下一个网络访问。

如果适用所有这些条件，您需要静态路由以在多个网络间获得正常运行的通信：

- 流量必须通过多个网络。
- 通过默认网关的独占流量流不足。

[需要静态路由的网络示例](#) 部分描述了场景，以及在没有配置静态路由时流量如何在不同的网络间流动。

### 25.1. 需要静态路由的网络示例

您需要在这个示例中的静态路由，因为并非所有 IP 网络都通过一个路由器直接连接。如果没有静态路由，一些网络无法相互通信。此外，某些网络流的流量仅有一个方向。



#### 注意

本例中的网络拓扑是假设的，仅用于解释静态路由的概念。在生产环境中并不推荐使用这个拓扑。

对于本示例中所有网络中的一个正常运行的通信，请将静态路由配置为 **Raleigh (198.51.100.0/24)**，下一个跃点 (hop) 路由器 2 (203.0.113.10)。下一个跃点的 IP 地址是数据中心网络中的一个路由器 2 (203.0.113.0/24)。

您可以配置静态路由，如下所示：

- 对于简化的配置，仅在路由器 1 上设置此静态路由。但是，这会增加路由器 1 上的流量，因为来自数据中心 (203.0.113.0/24) 的主机将流量发送到 Raleigh (198.51.100.0/24)，始终通过

## 路由器 1 到路由器 2。

- 对于更复杂的配置，请在数据中心的所有主机上配置此静态路由 (203.0.113.0/24)。然后，此子网中的所有主机直接向路由器 2 (203.0.113.10) 发送更接近 Raleigh 的主机 (198.51.100.0/24)。

有关网络流量流或不符的更多详情，请参见以下示意图中的说明。



在所需的静态路由没有配置的情况下，如下是通信可以正常工作和不能正常工作的情况：

- Berlin 网络中的主机 (192.0.2.0/24) :**

- 可以与同一子网中的其他主机通信，因为它们是直接连接的。

270\_RHEL\_0822

- 可以与互联网进行通信，因为路由器 1 位于 Berlin 网络(192.0.2.0/24)中，并且有默认网关，其可以访问互联网。
  - 可以与数据中心网络 (203.0.113.0/24)通信，因为路由器 1 在 Berlin (192.0.2.0/24) 和数据中心(203.0.113.0/24) 网络中都有接口。
  - 无法与 Raleigh 网络 (198.51.100.0/24) 通信，因为路由器 1 在此网络中没有接口。因此，路由器 1 将流量发送到自己的默认网关(互联网)。
- 数据中心网络中的主机 (203.0.113.0/24):
    - 可以与同一子网中的其他主机通信，因为它们是直接连接的。
    - 可以与互联网进行通信，因为它们有设置到路由器 1 的默认网关，路由器 1 在两个网络，数据中心(203.0.113.0/24)和互联网中都有接口。
    - 可以与 Berlin 网络 (192.0.2.0/24) 通信，因为它们的默认网关设置为路由器 1，并且路由器 1 在数据中心 (203.0.113.0/24) 和 Berlin (192.0.2.0/24) 网络中都存在接口。
    - 无法与 Raleigh 网络 (198.51.100.0/24) 通信，因为该网络中没有接口。因此，数据中心中的主机 (203.0.113.0/24) 将流量发送到其默认网关 (路由器 1)。路由器 1 在 Raleigh 网络 (198.51.100.0/24)中没有接口，因此路由器 1 将此流量发送到自己的默认网关(互联网)。
  - Raleigh 网络中的主机 (198.51.100.0/24) :
    - 可以与同一子网中的其他主机通信，因为它们是直接连接的。
    - 无法与互联网上的主机进行通信。因为默认的网关设置，路由器 2 将流量发送到路由器 1。路由器 1 的实际行为取决于反向路径过滤器 (rp\_filter) 系统控制 (sysctl) 设置。默认情况下，在 RHEL 上，路由器 1 会丢弃传出流量，而不是将其路由到互联网。但是，无论配置的行为如何，都无法在没有静态路由的情况下进行通信。

- 无法与数据中心网络通信(203.0.113.0/24)。由于默认网关设置，传出流量通过路由器 2 到达目的地。但是，对数据包的回复不会到达发送者，因为数据中心网络中的主机(203.0.113.0/24)将回复发送到其默认网关(Router 1)。然后，路由器 1 将流量发送到互联网。
- 无法与 Berlin 网络通信(192.0.2.0/24)。因为默认的网关设置，路由器 2 将流量发送到路由器 1。路由器 1 的实际行为取决于 sysctl 设置 rp\_filter。默认情况下，在 RHEL 中，路由器 1 会丢弃传出流量，而不是将其发送到 Berlin 网络(192.0.2.0/24)。但是，无论配置的行为如何，都无法在没有静态路由的情况下进行通信。



### 注意

除了配置静态路由外，还必须在两个路由器上启用 IP 转发。

## 其他资源

- [如果在服务器上设置了 net.ipv4.conf.all.rp\\_filter，为什么无法 ping 服务器？（红帽知识库）](#)
- [启用 IP 转发（红帽知识库）](#)

## 25.2. 如何使用 NMCLI 工具配置静态路由

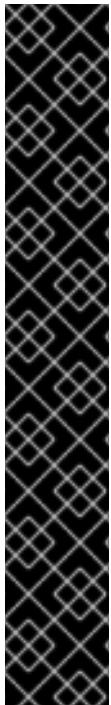
要配置静态路由，请使用具有以下语法的 nmcli 工具：

```
$ nmcli connection modify connection_name ipv4.routes "ip[/prefix] [next_hop] [metric]
[attribute=value] [attribute=value] ..."
```

该命令支持以下路由属性：

- **cwnd=n**：设置拥塞窗口(CWND)大小，以数据包数定义。
- **lock-cwnd=true|false**：定义内核是否可以更新 CWND 值。

- **lock-mtu=true|false** : 定义内核是否可以将 MTU 更新为路径 MTU 发现。
- **lock-window=true|false** : 定义内核是否可更新 TCP 数据包的最大窗口大小。
- **mtu=<mtu\_value>** : 设置沿目标路径使用的最大传输单元(MTU)。
- **onlink=true|false** : 定义下一个跃点是否直接附加到此链接，即使它与任何接口前缀都不匹配。
- **scope=<scope>** : 对于 IPv4 路由，此属性设置路由前缀涵盖的目标的范围。将值设为整数(0-255)。
- **src=<source\_address>** : 当将流量发送到路由前缀所涵盖的目标时，将源地址设置为首选。
- **table=<table\_id>** : 设置应将路由添加到的表的 ID。如果省略此参数，NetworkManager 将使用 main 表。
- **tos=<type\_of\_service\_key>** : 设置服务(TOS)密钥的类型。将值设为整数(0-255)。
- **type=<route\_type>** : 设置路由类型。NetworkManager 支持 unicast、local、blackhole、unreachable、prohibit 和 throw 路由类型。默认为 unicast。
- **window=<>window\_size>** : 设置 TCP 向这些目标通告的最大窗口大小，以字节为单位。



## 重要

如果您使用 `ipv4.routes` 选项，而前面没有 `+` 符号，则 `nmcli` 会覆盖这个参数的所有当前设置。

- 要创建额外的路由，请输入：

```
$ nmcli connection modify connection_name +ipv4.routes "<route>"
```

- 要删除特定的路由，请输入：

```
$ nmcli connection modify connection_name -ipv4.routes "<route>"
```

### 25.3. 使用 NMCLI 配置静态路由

您可以使用 `nmcli connection modify` 命令将静态路由添加到现有 NetworkManager 连接配置集。

以下流程配置以下路由：

- 到远程 198.51.100.0/24 网络的 IPv4 路由。IP 地址为 192.0.2.10 的相应网关可以通过 LAN 连接配置文件访问。
- 到远程 2001:db8:2::/64 网络的 IPv6 路由。IP 地址为 2001:db8:1::10 的相应网关可以通过 LAN 连接配置文件访问。

#### 先决条件

- LAN 连接配置文件存在，且其将位于同一个 IP 子网中的此主机配置为网关。

#### 流程

- 将静态 IPv4 路由添加到 LAN 连接配置文件中：

```
nmcli connection modify LAN +ipv4.routes "198.51.100.0/24 192.0.2.10"
```

要在多个步骤中设置多个路由，请将用逗号分开的单个路由传给命令：

```
nmcli connection modify <connection_profile> +ipv4.routes
"<remote_network_1>/<subnet_mask_1> <gateway_1>,
<remote_network_n>/<subnet_mask_n> <gateway_n>, ..."
```

2.

将静态 IPv6 路由添加到 LAN 连接配置文件中：

```
nmcli connection modify LAN +ipv6.routes "2001:db8:2::/64 2001:db8:1::10"
```

3.

重新激活连接：

```
nmcli connection up LAN
```

## 验证

1.

显示 IPv4 路由：

```
ip -4 route
...
198.51.100.0/24 via 192.0.2.10 dev enp1s0
```

2.

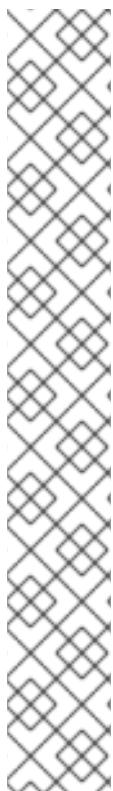
显示 IPv6 路由：

```
ip -6 route
...
2001:db8:2::/64 via 2001:db8:1::10 dev enp1s0 metric 1024 pref medium
```

## 25.4. 使用 NMTUI 配置静态路由

nmtui 应用程序为 NetworkManager 提供了一个基于文本的用户界面。您可以使用 nmtui 在没有图形界面的主机上配置静态路由。

例如，以下流程将路由添加到 192.0.2.0/24 网络，该网络使用运行在 198.51.100.1 上的网关，该网络可通过现有的连接配置文件访问。



## 注意

在 nmtui 中：

- 使用光标键导航。
- 选择一个按钮并按 Enter 键。
- 使用 空格 选择和清除复选框。
- 要返回上一个屏幕，请使用 ESC。

## 先决条件

- 网络已配置。
- 静态路由的网关必须在接口上直接访问。
- 如果用户在物理控制台中登录，用户权限就足够了。否则，命令需要 root 权限。

## 流程

1.

启动 nmtui：

```
nmtui
```

2.

选择 Edit a connection，然后按 Enter。

3.

选择您可通过其到达目的地网络的下一跳的连接配置文件，然后按 Enter。

4.

根据它是 IPv4 路由还是 IPv6 路由，按协议配置区域旁边的 Show 按钮。

5.

按 Routing 旁边的 Edit 按钮。这将打开一个新窗口，您可在其中配置静态路由：

a.

按 Add 按钮并填写：

- 目的网络，包括无类别域间路由(CIDR)格式的前缀
- 下一跳的 IP 地址
- 指标值，如果您向同一网络添加多个路由，并且希望根据效率对路由进行优先排序

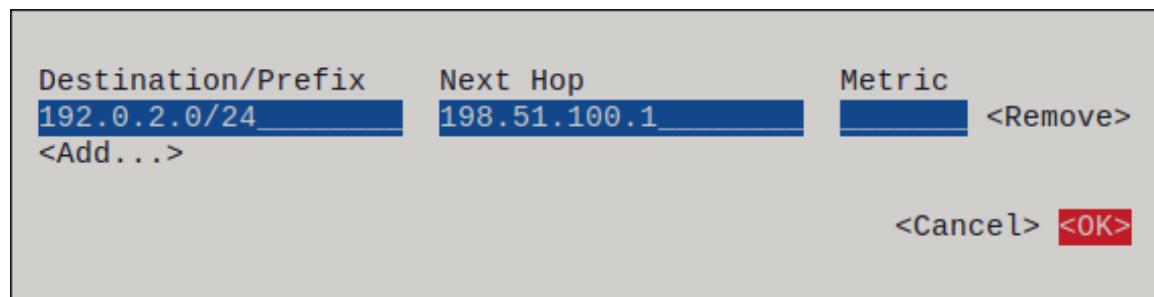
b.

对您要添加的每个路由重复上一步，且通过此连接配置文件可达。

c.

按 OK 按钮返回到连接设置窗口。

图 25.1. 没有指标的静态路由的示例



6.

按 OK 按钮返回到 nmtui 主菜单。

7.

选择 Activate a connection，然后按 Enter。

8.

选择您编辑的连接配置文件，然后按 Enter 两次来停用并再次激活它。

**重要**

如果您通过使用您要重新激活的连接配置文件的远程连接（如 SSH）来运行 nmtui，请跳过这一步。在这种情况下，如果您在 nmtui 中停用了它，连接将被终止，因此您无法再次激活它。要避免这个问题，请使用 nmcli connection <connection\_profile> up 命令，来在上述场景中重新激活连接。

9. 按 Back 按钮返回到主菜单。
  
10. 选择 Quit，然后按 Enter 键关闭 nmtui 应用程序。

**验证**

- 验证路由是否处于活跃状态：

```
$ ip route
...
192.0.2.0/24 via 198.51.100.1 dev example proto static metric 100
```

**25.5. 使用 CONTROL-CENTER 配置静态路由**

您可以在 GNOME 中使用 control-center，来将静态路由添加到网络连接配置中。

以下流程配置以下路由：

- 到远程 198.51.100.0/24 网络的 IPv4 路由。对应的网关的 IP 地址 192.0.2.10。
  
- 到远程 2001:db8:2::/64 网络的 IPv6 路由。对应的网关具有 IP 地址 2001:db8:1::10。

**前提条件**

- 网络已配置。

- 此主机与网关位于同一个 IP 子网中。
- 连接的网络配置在 **control-center** 应用程序中打开。请参阅 [使用 nm-connection-editor 配置以太网连接](#)。

## 流程

1.

在 IPv4 标签页中：

a.

可选：点 IPv4 选项卡的 **Routes** 部分中的 **On** 按钮来禁用自动路由，使其只使用静态路由。如果启用了自动路由，Red Hat Enterprise Linux 将使用静态路由和从 DHCP 服务器接收的路由。

b.

输入 IPv4 路由的地址、子网掩码、网关和可选的指标值：

<b>Routes</b>				<b>Automatic</b> <input checked="" type="checkbox"/>
<b>Address</b>	<b>Netmask</b>	<b>Gateway</b>	<b>Metric</b>	
198.51.100.0	24	192.0.2.10		<input checked="" type="checkbox"/>

2.

在 IPv6 标签页中：

a.

可选：点 IPv4 选项卡的 **Routes** 部分中的 **On** 按钮来禁用自动路由，使其只使用静态路由。

b.

输入 IPv6 路由的地址、子网掩码、网关和可选的指标值：

<b>Routes</b>				<b>Automatic</b> <input checked="" type="checkbox"/>
<b>Address</b>	<b>Prefix</b>	<b>Gateway</b>	<b>Metric</b>	
2001:db8:2::	64	2001:db8:1::10		<input checked="" type="checkbox"/>

3.

点应用。

4.

返回到 Network 窗口，通过将连接的按钮切换为 Off，然后返回到 On 来禁用并重新启用连接，以使更改生效。



## 验证

1.

显示 IPv4 路由：

```
ip -4 route
...
198.51.100.0/24 via 192.0.2.10 dev enp1s0
```

2.

显示 IPv6 路由：

```
ip -6 route
...
2001:db8:2::/64 via 2001:db8:1::10 dev enp1s0 metric 1024 pref medium
```

## 25.6. 使用 NM-CONNECTION-EDITOR 配置静态路由

您可以使用 `nm-connection-editor` 应用程序将静态路由添加到网络连接配置中。

以下流程配置以下路由：

- 

到远程 198.51.100.0/24 网络的 IPv4 路由。IP 地址为 192.0.2.10 的对应网关可以通过

**example** 连接访问。

- 到远程 2001:db8:2::/64 网络的 IPv6 路由。IP 地址为 2001:db8:1::10 的对应网关可以通过 **example** 连接访问。

## 前提条件

- 网络已配置。
- 此主机与网关位于同一个 IP 子网中。

## 流程

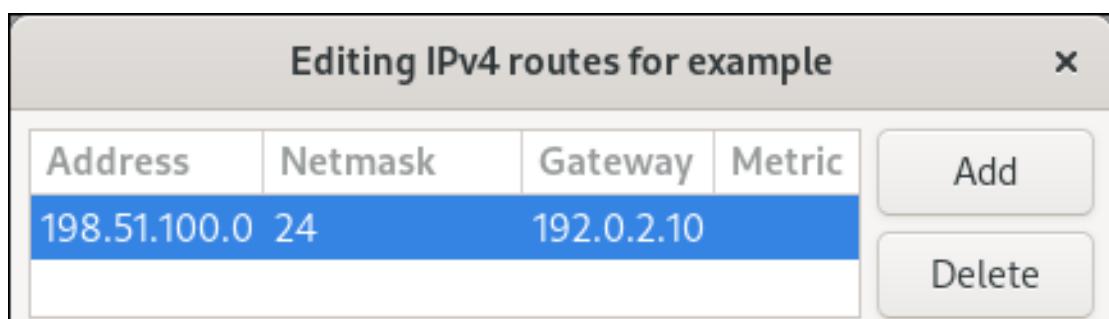
1. 打开一个终端，输入 **nm-connection-editor**：

```
$ nm-connection-editor
```

2. 选择 **example** 连接配置集，并点齿轮图标编辑现有连接。

3. 在 IPv4 设置 标签页中：

- a. 点击 路由 按钮。
- b. 点击 添加 按钮并输入地址、子网掩码、网关以及可选的指标值。



C.

点击 确定。

4.

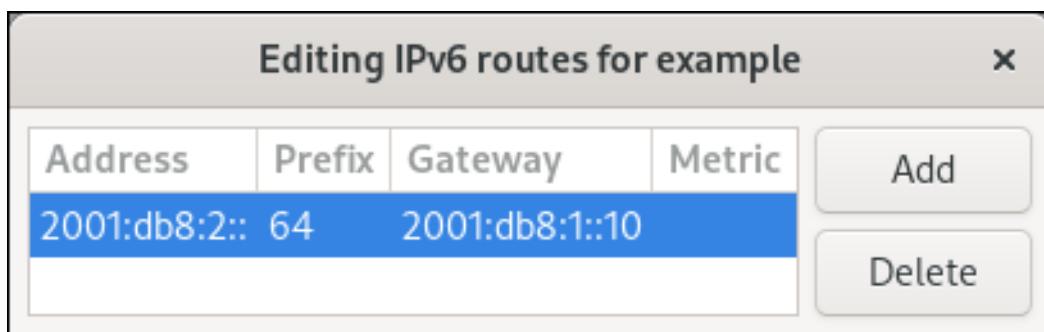
在 IPv6 设置 标签页中：

a.

点击 路由 按钮。

b.

点击 添加 按钮并输入地址、子网掩码、网关以及可选的指标值。



C.

点击 确定。

5.

点击 Save。

6.

重启网络连接以使更改生效。例如，要使用命令行重启 example 连接：

```
nmcli connection up example
```

验证

1.

显示 IPv4 路由：

```
ip -4 route
...
198.51.100.0/24 via 192.0.2.10 dev enp1s0
```

2.

显示 IPv6 路由：

```
ip -6 route
...
2001:db8:2::/64 via 2001:db8:1::10 dev enp1s0 metric 1024 pref medium
```

## 25.7. 使用 NMCLI 交互模式配置静态路由

您可以使用 nmcli 工具的交互模式，将静态路由添加到网络连接配置中。

以下流程配置以下路由：

- 到远程 198.51.100.0/24 网络的 IPv4 路由。IP 地址为 192.0.2.10 的对应网关可以通过 example 连接访问。
- 到远程 2001:db8:2::/64 网络的 IPv6 路由。IP 地址为 2001:db8:1::10 的对应网关可以通过 example 连接访问。

### 前提条件

- example 连接配置集存在，它将此主机配置为与网关在同一 IP 子网中。

### 流程

1.

为 example 连接打开 nmcli 交互模式：

```
nmcli connection edit example
```

2.

添加静态 IPv4 路由：

```
nmcli> set ipv4.routes 198.51.100.0/24 192.0.2.10
```

3.

添加静态 IPv6 路由：

```
nmcli> set ipv6.routes 2001:db8:2::/64 2001:db8:1::10
```

4.

可选：验证路由是否被正确添加到配置中：

```
nmcli> print
...
ipv4.routes: { ip = 198.51.100.0/24, nh = 192.0.2.10 }
...
ipv6.routes: { ip = 2001:db8:2::/64, nh = 2001:db8:1::10 }
...
```

ip 属性显示要路由的网络，nh 属性显示网关（下一跳）。

5.

保存配置：

```
nmcli> save persistent
```

6.

重启网络连接：

```
nmcli> activate example
```

7.

保留 nmcli 交互模式：

```
nmcli> quit
```

验证

1.

显示 IPv4 路由：

```
ip -4 route
...
198.51.100.0/24 via 192.0.2.10 dev enp1s0
```

2.

显示 IPv6 路由：

```
ip -6 route
...
2001:db8:2::/64 via 2001:db8:1::10 dev enp1s0 metric 1024 pref medium
```

## 其他资源

- 您系统上的 **nmcli** (1) 和 **nm-settings-nmcli** (5) 手册页

## 25.8. 使用 NMSTATECTL 配置静态路由

使用 **nmstatectl** 工具，通过 **Nmstate API** 配置静态路由。Nmstate API 确保设置配置后结果与配置文件匹配。如果有任何失败，**nmstatectl** 会自动回滚更改以避免系统处于不正确的状态。

### 先决条件

- enp1s0** 网络接口已配置，且与网关位于同一个 IP 子网。
- nmstate** 软件包已安装。

### 流程

- 创建一个包含以下内容的 YAML 文件，如 `~/add-static-route-to-enp1s0.yml`：

```

routes:
 config:
 - destination: 198.51.100.0/24
 next-hop-address: 192.0.2.10
 next-hop-interface: enp1s0
 - destination: 2001:db8:2::/64
 next-hop-address: 2001:db8:1::10
 next-hop-interface: enp1s0
```

这些设置定义以下静态路由：

- 到远程 198.51.100.0/24 网络的 IPv4 路由。IP 地址为 192.0.2.10 的对应网关可以通过 **enp1s0** 接口访问。

- 到远程 2001:db8:2::/64 网络的 IPv6 路由。IP 地址为 2001:db8:1::10 的对应网关可以通过 `enp1s0` 接口访问。
2. 将设置应用到系统：

```
nmstatectl apply ~/add-static-route-to-enp1s0.yml
```

## 验证

1. 显示 IPv4 路由：

```
ip -4 route
...
198.51.100.0/24 via 192.0.2.10 dev enp1s0
```

2. 显示 IPv6 路由：

```
ip -6 route
...
2001:db8:2::/64 via 2001:db8:1::10 dev enp1s0 metric 1024 pref medium
```

## 其他资源

- 您系统上的 `nmstatectl` (8) 手册页
- `/usr/share/doc/nmstate/examples/` 目录

## 25.9. 使用 NETWORK RHEL 系统角色配置静态路由

静态路由确保您可以将流量发送到无法通过默认网关到达的目的地。您可以在连接到相同网络的接口的 NetworkManager 连接配置文件中，将静态路由配置为下一跳。通过使用 Ansible 和 network RHEL 系统角色，您可以自动化此过程，并在 playbook 中定义的主机上远程配置连接配置文件。



### 警告

您不能使用 **network RHEL** 系统角色只更新现有连接配置文件中的特定值。该角色确保连接配置文件与 **playbook** 中的设置完全匹配。如果具有相同名称的连接配置文件存在，则角色将应用 **playbook** 中的设置，并将配置文件中的所有其他设置重置为其默认值。要防止重置值，请始终在 **playbook** 中指定网络连接配置文件的整个配置，包括您不想更改的设置。

## 先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 **playbook** 的用户登录到控制节点。
- 您用于连接到受管节点的帐户对它们具有 **sudo** 权限。

## 流程

1.

创建一个包含以下内容的 **playbook** 文件，如 `~/playbook.yml`：

```

- name: Configure the network
 hosts: managed-node-01.example.com
 tasks:
 - name: Ethernet connection profile with static IP address settings
 ansible.builtin.include_role:
 name: redhat.rhel_system_roles.network
 vars:
 network_connections:
 - name: enp7s0
 type: ethernet
 autoconnect: yes
 ip:
 address:
 - 192.0.2.1/24
 - 2001:db8:1::1/64
 gateway4: 192.0.2.254
 gateway6: 2001:db8:1::ffffe
 dns:
 - 192.0.2.200
```

```

- 2001:db8:1::ffbb
dns_search:
- example.com
route:
- network: 198.51.100.0
prefix: 24
gateway: 192.0.2.10
- network: '2001:db8:2::'
prefix: 64
gateway: 2001:db8:1::10
state: up

```

有关 playbook 中使用的所有变量的详情，请查看控制节点上的  
`/usr/share/ansible/roles/rhel-system-roles.network/README.md` 文件。

2.

验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不能防止错误的、但有效的配置。

3.

运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

1.

显示 IPv4 路由：

```
ansible managed-node-01.example.com -m command -a 'ip -4 route'
managed-node-01.example.com | CHANGED | rc=0 >>
...
198.51.100.0/24 via 192.0.2.10 dev enp7s0
```

2.

显示 IPv6 路由：

```
ansible managed-node-01.example.com -m command -a 'ip -6 route'
managed-node-01.example.com | CHANGED | rc=0 >>
...
2001:db8:2::/64 via 2001:db8:1::10 dev enp7s0 metric 1024 pref medium
```

## 其他资源

- `/usr/share/ansible/roles/rhel-system-roles.network/README.md` 文件
- `/usr/share/doc/rhel-system-roles/network/` 目录

### 25.10. 使用旧的网络脚本以键-值格式创建静态路由配置文件

旧的网络脚本支持以 **key-value** 格式设置静态路由。

以下流程向远程 198.51.100.0/24 网络配置一个 IPv4 路由。IP 地址为 192.0.2.10 的对应网关可以通过 `enp1s0` 接口访问。



#### 注意

旧的网络脚本只支持静态 IPv4 路由的键值格式。对于 IPv6 路由，请使用 `ip` 命令格式。请参阅[在使用旧的网络脚本时，通过 ip-command-format 来创建静态路由配置文件](#)。

#### 前提条件

- 静态路由的网关必须在接口上直接访问。
- `NetworkManager` 软件包未安装，或者 `NetworkManager` 服务被禁用。
- `network-scripts` 软件包已安装。
- `network` 服务已启用。

#### 流程

1. 将静态 IPv4 路由添加到 `/etc/sysconfig/network-scripts/route-enp0s1` 文件中：

```
ADDRESS0=198.51.100.0
NETMASK0=255.255.255.0
GATEWAY0=192.0.2.10
```

- **ADDRESS0** 变量定义第一个路由条目的网络。
- **NETMASK0** 变量定义第一个路由条目的子网掩码。
- **GATEWAY0** 变量定义到远程网络的网关的 IP 地址或第一个路由条目的主机。

如果您添加多个静态路由，请增加变量名称的数量。请注意，每个路由的变量都必须按顺序编号。例如，**ADDRESS0**、**ADDRESS1**、**ADDRESS3**，等等。

2.

重启网络：

```
systemctl restart network
```

验证

●

显示 IPv4 路由：

```
ip -4 route
...
198.51.100.0/24 via 192.0.2.10 dev enp1s0
```

故障排除

●

显示 network 单元的日志条目：

```
journalctl -u network
```

以下是可能的错误信息及其原因：

○

**Error: Nexthop has invalid gateway:** 在 route-enp1s0 文件中指定一个没有与此路由器相同的子网中的 IPv4 网关地址。

- RTNETLINK answers: No route to host: 在 route6-*enp1s0* 文件中指定一个没有与此路由器相同的子网中的 IPv6 网关地址。
- Error: Invalid prefix for given prefix length : 您可以使用远程网络中的 IP 地址（而非网络地址）在 route-*enp1s0* 文件中指定远程网络。

## 其他资源

- /usr/share/doc/network-scripts/sysconfig.txt 文件

### 25.11. 在使用旧的网络脚本时，使用 IP-COMMAND-FORMAT 创建静态路由配置文件

旧的网络脚本支持设置静态路由。

以下流程配置以下路由：

- 到远程 198.51.100.0/24 网络的 IPv4 路由。IP 地址为 192.0.2.10 的对应网关可以通过 *enp1s0* 接口访问。
- 到远程 2001:db8:2::/64 网络的 IPv6 路由。IP 地址为 2001:db8:1::10 的对应网关可以通过 *enp1s0* 接口访问。



#### 重要

网关（下一跳）的 IP 地址必须与您在其上配置静态路由的主机位于同一个 IP 子网中。

此流程中的示例使用 ip-command 格式的配置条目。

## 前提条件

- 静态路由的网关必须在接口上直接访问。

- NetworkManager 软件包未安装，或者 NetworkManager 服务被禁用。
- network-scripts 软件包已安装。
- network 服务已启用。

## 流程

1. 将静态 IPv4 路由添加到 /etc/sysconfig/network-scripts/route-enp1s0 文件中：

```
198.51.100.0/24 via 192.0.2.10 dev enp1s0
```

始终指定远程网络的网络地址，如 198.51.100.0。在远程网络中设置 IP 地址，如 198.51.100.1 会导致网络脚本添加此路由。

2. 将静态 IPv6 路由添加到 /etc/sysconfig/network-scripts/route6-enp1s0 文件中：

```
2001:db8:2::/64 via 2001:db8:1::10 dev enp1s0
```

3. 重启 network 服务：

```
systemctl restart network
```

## 验证

1. 显示 IPv4 路由：

```
ip -4 route
...
198.51.100.0/24 via 192.0.2.10 dev enp1s0
```

2. 显示 IPv6 路由：

```
ip -6 route
...
2001:db8:2::/64 via 2001:db8:1::10 dev enp1s0 metric 1024 pref medium
```

## 故障排除

- 显示 network 单元的日志条目：

```
journalctl -u network
```

以下是可能的错误信息及其原因：

- **Error: Nexthop has invalid gateway:** 在 route-enp1s0 文件中指定一个没有与此路由器相同的子网中的 IPv4 网关地址。
- **RTNETLINK answers: No route to host:** 在 route6-enp1s0 文件中指定一个没有与此路由器相同的子网中的 IPv6 网关地址。
- **Error: Invalid prefix for given prefix length :** 您可以使用远程网络中的 IP 地址（而非网络地址）在 route-enp1s0 文件中指定远程网络。

## 其他资源

- [/usr/share/doc/network-scripts/sysconfig.txt](#) 文件

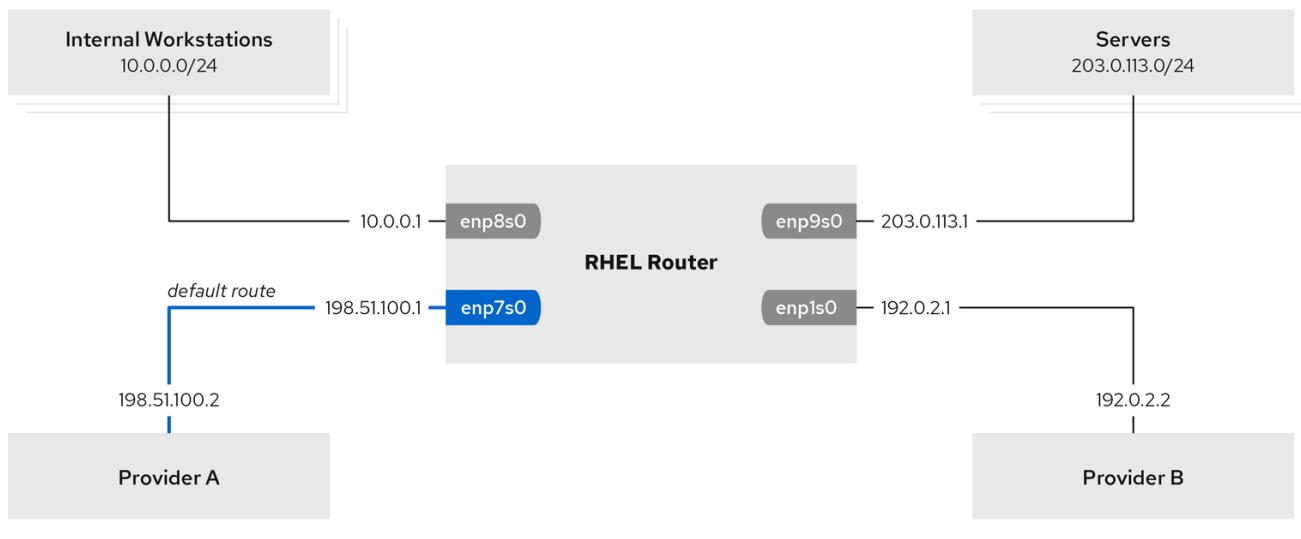
## 第 26 章 配置基于策略的路由以定义其他路由

默认情况下，RHEL 中的内核决定使用路由表根据目标地址转发网络数据包。基于策略的路由允许您配置复杂的路由场景。例如，您可以根据各种条件来路由数据包，如源地址、数据包元数据或协议。

### 26.1. 使用 NMCLI 将流量从特定子网路由到不同的默认网关

您可以使用基于策略的路由为来自特定子网的流量配置不同的默认网关。例如，您可以将 RHEL 配置为默认路由，使用默认路由将所有流量路由到互联网提供商 A。但是，从内部工作站子网接收的流量路由到供应商 B。

该流程假设以下网络拓扑：



60\_RHEL\_0120

### 先决条件

- 系统使用 **NetworkManager** 来配置网络，这是默认设置。
- 要在流程中设置的 RHEL 路由器有四个网络接口：
  - **enp7s0** 接口已连接到提供商 A 的网络。提供商网络中的网关 IP 为 198.51.100.2，网络

使用 /30 网络掩码。

- **enp1s0** 接口已连接到提供商 B 的网络。提供商网络中的网关 IP 为 192.0.2.2，网络使用 /30 网络掩码。
- **enp8s0** 接口已与连有内部工作站的 10.0.0.0/24 子网相连。
- **enp9s0** 接口已与连有公司服务器的 203.0.113.0/24 子网相连。
- 内部工作站子网中的主机使用 10.0.0.1 作为默认网关。在此流程中，您可以将这个 IP 地址分配给路由器的 **enp8s0** 网络接口。
- 服务器子网中的主机使用 203.0.113.1 作为默认网关。在此流程中，您可以将这个 IP 地址分配给路由器的 **enp9s0** 网络接口。
- **firewalld** 服务已启用，并处于活动状态。

## 流程

1. 将网络接口配置为供应商 A：

```
nmcli connection add type ethernet con-name Provider-A ifname enp7s0
ipv4.method manual ipv4.addresses 198.51.100.1/30 ipv4.gateway 198.51.100.2
ipv4.dns 198.51.100.200 connection.zone external
```

**nmcli connection add** 命令创建 **NetworkManager** 连接配置文件。该命令使用以下选项：

- **type ethernet**：定义连接类型是 Ethernet。
- **con-name <connection\_name>**：设置配置集的名称。使用有意义的名称以避免混淆。

- **ifname <network\_device>** : 设置网络接口。
- **ipv4.method manual** : 允许配置静态 IP 地址。
- **ipv4.addresses &lt;IP\_address&ampgt / <subnet\_mask>** : 设置 IPv4 地址和子网掩码。
- **ipv4.gateway <IP\_address>** : 设置默认网关地址。
- **ipv4.dns <IP\_of\_DNS\_server>** : 设置 DNS 服务器的 IPv4 地址。
- **connection.zone <firewalld\_zone>** : 将网络接口分配给定义的 firewalld 区。请注意, firewalld 会为分配给 external 区域的接口自动启用伪装。

2.

将网络接口配置为供应商 B:

```
nmcli connection add type ethernet con-name Provider-B ifname enp1s0
ipv4.method manual ipv4.addresses 192.0.2.1/30 ipv4.routes "0.0.0.0/0 192.0.2.2
table=5000" connection.zone external
```

此命令使用 **ipv4.routes** 参数而不是 **ipv4.gateway** 来设置默认网关。这需要将这个连接的默认网关分配给不同于默认的路由表(5000)。当连接被激活时, NetworkManager 会自动创建这个新的路由表。

3.

将网络接口配置为内部工作站子网 :

```
nmcli connection add type ethernet con-name Internal-Workstations ifname enp8s0
ipv4.method manual ipv4.addresses 10.0.0.1/24 ipv4.routes "10.0.0.0/24 table=5000"
ipv4.routing-rules "priority 5 from 10.0.0.0/24 table 5000" connection.zone trusted
```

此命令使用 **ipv4.routes** 参数将静态路由添加到 ID 为 5000 的路由表中。10.0.0.0/24 子网的这个静态路由使用到供应商 B 的本地网络接口的 IP 地址(192.0.2.1)来作为下一跳。

另外，命令使用 `ipv4.routing-rules` 参数来添加优先级为 5 的路由规则，该规则将来自 `10.0.0.0/24` 子网的流量路由到表 5000。低的值具有更高的优先级。

请注意，`ipv4.routing-rules` 参数的语法与 `ip rule add` 命令中的语法相同，但 `ipv4.routing-rules` 总是需要指定优先级。

4.

将网络接口配置为服务器子网：

```
nmcli connection add type ethernet con-name Servers ifname enp9s0 ipv4.method manual ipv4.addresses 203.0.113.1/24 connection.zone trusted
```

验证

1.

在内部工作站子网的 RHEL 主机上：

a.

安装 traceroute 软件包：

```
yum install traceroute
```

b.

使用 traceroute 工具显示到互联网上主机的路由：

```
traceroute redhat.com
traceroute to redhat.com (209.132.183.105), 30 hops max, 60 byte packets
 1 10.0.0.1 (10.0.0.1) 0.337 ms 0.260 ms 0.223 ms
 2 192.0.2.2 (192.0.2.2) 0.884 ms 1.066 ms 1.248 ms
...
```

命令的输出显示路由器通过 `192.0.2.1`，即提供商 B 的网络来发送数据包。

2.

在服务器子网的 RHEL 主机上：

a.

安装 traceroute 软件包：

```
yum install traceroute
```

b.

使用 traceroute 工具显示到互联网上主机的路由：

```
traceroute redhat.com
traceroute to redhat.com (209.132.183.105), 30 hops max, 60 byte packets
 1 203.0.113.1 (203.0.113.1) 2.179 ms 2.073 ms 1.944 ms
 2 198.51.100.2 (198.51.100.2) 1.868 ms 1.798 ms 1.549 ms
...
```

命令的输出显示路由器通过 198.51.100.2，即供应商 A 的网络来发送数据包。

## 故障排除步骤

在 RHEL 路由器中：

1.

显示规则列表：

```
ip rule list
0: from all lookup local
5: from 10.0.0.0/24 lookup 5000
32766: from all lookup main
32767: from all lookup default
```

默认情况下，RHEL 包含表 local、main 和 default 的规则。

2.

显示表 5000 中的路由：

```
ip route list table 5000
0.0.0.0/0 via 192.0.2.2 dev enp1s0 proto static metric 100
10.0.0.0/24 dev enp8s0 proto static scope link src 192.0.2.1 metric 102
```

3.

显示接口和防火墙区：

```
firewall-cmd --get-active-zones
external
 interfaces: enp1s0 enp7s0
trusted
 interfaces: enp8s0 enp9s0
```

4.

验证 **external** 区是否启用了伪装：

```
firewall-cmd --info-zone=external
external (active)
target: default
icmp-block-inversion: no
interfaces: enp1s0 enp7s0
sources:
services: ssh
ports:
protocols:
masquerade: yes
...
```

## 其他资源

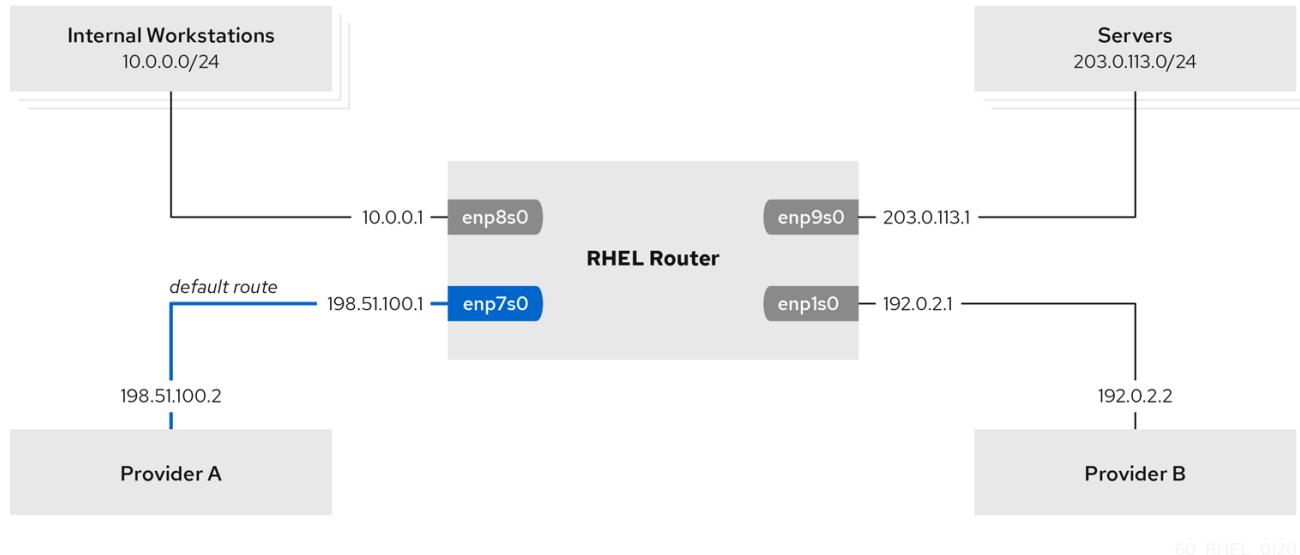
- 您系统上的 **nmcli** (1) 和 **nm-settings** (5) 手册页

## 26.2. 使用 NETWORK RHEL 系统角色将流量从特定子网路由到不同的默认网关

您可以使用基于策略的路由为来自特定子网的流量配置不同的默认网关。例如，您可以将 RHEL 配置为默认路由，使用默认路由将所有流量路由到互联网提供商 A。但是，从内部工作站子网收到的流量被路由到提供者 B。通过使用 Ansible 和 **network RHEL** 系统角色，您可以自动化此过程，并在 **playbook** 中定义的主机上远程配置连接配置文件。

您可以使用 **network RHEL** 系统角色配置连接配置文件，包括路由表和规则。

此流程假设以下网络拓扑：



## 先决条件

- 您已准备好控制节点和受管节点。
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 您用于连接到受管节点的帐户对它们具有 sudo 权限。
- 受管节点使用 NetworkManager 和 firewalld 服务。
- 您要配置的受管节点有 4 个网络接口：
  - enp7s0** 接口已连接到提供商 A 的网络。提供商网络中的网关 IP 为 198.51.100.2，网络使用 /30 网络掩码。
  - enp1s0** 接口已连接到提供商 B 的网络。提供商网络中的网关 IP 为 192.0.2.2，网络使用 /30 网络掩码。

- enp8s0 接口已与连有内部工作站的 10.0.0.0/24 子网相连。
- enp9s0 接口已与连有公司服务器的 203.0.113.0/24 子网相连。
- 内部工作站子网中的主机使用 10.0.0.1 作为默认网关。在此流程中，您可以将这个 IP 地址分配给路由器的 enp8s0 网络接口。
- 服务器子网中的主机使用 203.0.113.1 作为默认网关。在此流程中，您可以将这个 IP 地址分配给路由器的 enp9s0 网络接口。

## 流程

1.

创建一个包含以下内容的 playbook 文件，如 ~/playbook.yml：

```

- name: Configuring policy-based routing
 hosts: managed-node-01.example.com
 tasks:
 - name: Routing traffic from a specific subnet to a different default gateway
 ansible.builtin.include_role:
 name: redhat.rhel_system_roles.network
 vars:
 network_connections:
 - name: Provider-A
 interface_name: enp7s0
 type: ethernet
 autoconnect: True
 ip:
 address:
 - 198.51.100.1/30
 gateway4: 198.51.100.2
 dns:
 - 198.51.100.200
 state: up
 zone: external

 - name: Provider-B
 interface_name: enp1s0
 type: ethernet
 autoconnect: True
 ip:
 address:
 - 192.0.2.1/30
 route:
 - network: 0.0.0.0
```

```

prefix: 0
gateway: 192.0.2.2
table: 5000
state: up
zone: external

- name: Internal-Workstations
 interface_name: enp8s0
 type: ethernet
 autoconnect: True
 ip:
 address:
 - 10.0.0.1/24
 route:
 - network: 10.0.0.0
 prefix: 24
 table: 5000
 routing_rule:
 - priority: 5
 from: 10.0.0.0/24
 table: 5000
 state: up
 zone: trusted

- name: Servers
 interface_name: enp9s0
 type: ethernet
 autoconnect: True
 ip:
 address:
 - 203.0.113.1/24
 state: up
 zone: trusted

```

示例 playbook 中指定的设置包括如下：

**table: <value>**

将来自同一列表条目中的路由作为 table 变量分配给指定的路由表。

**routing\_rule: <list>**

定义指定的路由规则的优先级，以及从连接配置文件到分配了规则的路由表的优先级。

**zone: <zone\_name>**

将连接配置文件中的网络接口分配给指定的 firewalld 区域。

有关 playbook 中使用的所有变量的详情，请查看控制节点上的

/usr/share/ansible/roles/rhel-system-roles.network/README.md 文件。

2.

验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不能防止错误的、但有效的配置。

3.

运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

1.

在内部工作站子网的 RHEL 主机上：

a.

安装 traceroute 软件包：

```
yum install traceroute
```

b.

使用 traceroute 工具显示到互联网上主机的路由：

```
traceroute redhat.com
traceroute to redhat.com (209.132.183.105), 30 hops max, 60 byte packets
 1 10.0.0.1 (10.0.0.1) 0.337 ms 0.260 ms 0.223 ms
 2 192.0.2.2 (192.0.2.2) 0.884 ms 1.066 ms 1.248 ms
...
```

命令的输出显示路由器通过 192.0.2.1，即提供商 B 的网络来发送数据包。

2.

在服务器子网的 RHEL 主机上：

a.

安装 traceroute 软件包：

```
yum install traceroute
```

b.

使用 traceroute 工具显示到互联网上主机的路由：

```
traceroute redhat.com
traceroute to redhat.com (209.132.183.105), 30 hops max, 60 byte packets
1 203.0.113.1 (203.0.113.1) 2.179 ms 2.073 ms 1.944 ms
2 198.51.100.2 (198.51.100.2) 1.868 ms 1.798 ms 1.549 ms
...
...
```

命令的输出显示路由器通过 198.51.100.2，即供应商 A 的网络来发送数据包。

3.

在使用 RHEL 系统角色配置的 RHEL 路由器上：

a.

显示规则列表：

```
ip rule list
0: from all lookup local
5: from 10.0.0.0/24 lookup 5000
32766: from all lookup main
32767: from all lookup default
```

默认情况下，RHEL 包含表 local、main 和 default 的规则。

b.

显示表 5000 中的路由：

```
ip route list table 5000
0.0.0.0/0 via 192.0.2.2 dev enp1s0 proto static metric 100
10.0.0.0/24 dev enp8s0 proto static scope link src 192.0.2.1 metric 102
```

c.

显示接口和防火墙区：

```
firewall-cmd --get-active-zones
external
interfaces: enp1s0 enp7s0
trusted
interfaces: enp8s0 enp9s0
```

d.

验证 **external** 区是否启用了伪装：

```
firewall-cmd --info-zone=external
external (active)
target: default
icmp-block-inversion: no
interfaces: enp1s0 enp7s0
sources:
services: ssh
ports:
protocols:
masquerade: yes
...
```

## 其他资源

- `/usr/share/ansible/roles/rhel-system-roles.network/README.md` 文件
- `/usr/share/doc/rhel-system-roles/network/` 目录

### 26.3. 使用旧网络脚本时，涉及基于策略的路由的配置文件概述

如果您使用旧的网络脚本而不是 NetworkManager 配置网络，您也可以配置基于策略的路由。



#### 注意

使用 `network-scripts` 软件包提供的旧网络脚本来配置网络已在 RHEL 8 中被弃用。使用 NetworkManager 配置基于策略的路由。例如，请参阅 [使用 nmcli 将来自特定子网的流量路由到不同的默认网关](#)。

使用旧的网络脚本时，以下配置文件会涉及基于策略的路由：

- `/etc/sysconfig/network-scripts/route-interface`：此文件定义 IPv4 路由。使用 `table` 选项来指定路由表。例如：

```
192.0.2.0/24 via 198.51.100.1 table 1
203.0.113.0/24 via 198.51.100.2 table 2
```

- `/etc/sysconfig/network-scripts/route6-interface` : 此文件定义 IPv6 路由。
- `/etc/sysconfig/network-scripts/rule-interface` : 此文件定义内核将流量路由到特定路由表的 IPv4 源网络的规则。例如：
 

```
from 192.0.2.0/24 lookup 1
from 203.0.113.0/24 lookup 2
```
- `/etc/sysconfig/network-scripts/rule6-interface` : 此文件定义内核将流量路由到特定路由表的 IPv6 源网络的规则。
- `/etc/iproute2/rt_tables` : 如果您想要使用名称而不是数字来引用特定的路由表，这个文件定义了映射。例如：
 

```
1 Provider_A
2 Provider_B
```

## 其他资源

- 您系统上的 `ip-route (8)` 和 `ip-rule (8)` 手册页

### 26.4. 使用旧的网络脚本将来自特定子网的流量路由到不同的默认网关

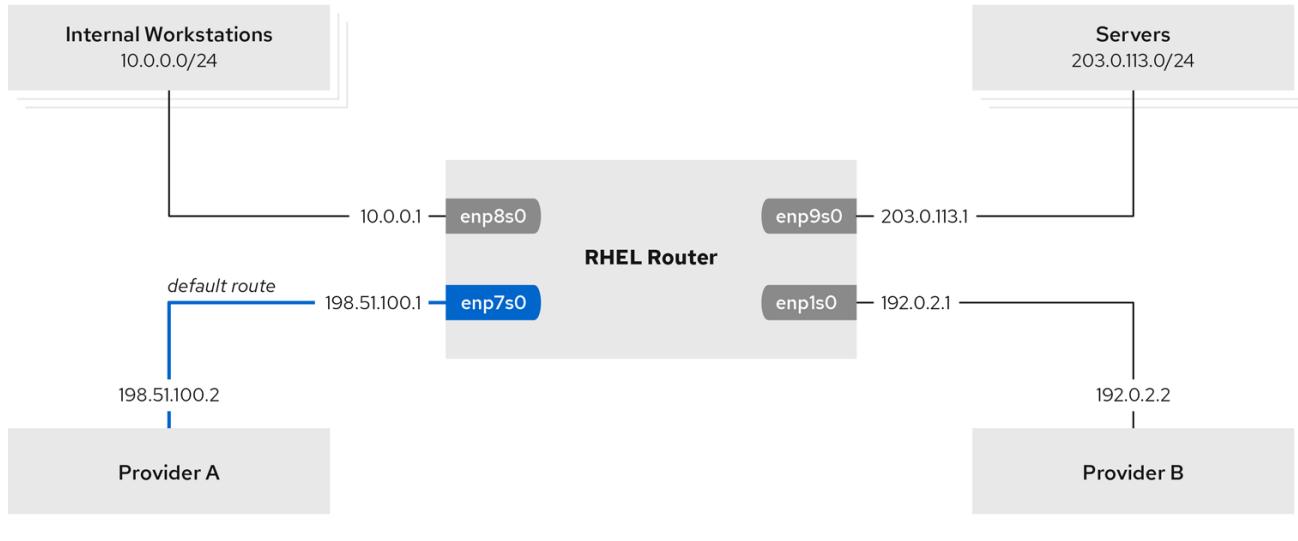
您可以使用基于策略的路由为来自特定子网的流量配置不同的默认网关。例如，您可以将 RHEL 配置为默认路由，使用默认路由将所有流量路由到互联网提供商 A。但是，从内部工作站子网接收的流量路由到供应商 B。



#### 重要

使用 `network-scripts` 软件包提供的旧网络脚本来配置网络已在 RHEL 8 中被弃用。只有您在主机上使用旧的网络脚本而不是 `NetworkManager` 时，才按照以下流程操作。如果您使用 `NetworkManager` 管理网络设置，请参阅 [使用 nmcli 将来自特定子网的流量路由到不同的默认网关](#)。

该流程假设以下网络拓扑：



60\_RHEL\_0120



### 注意

旧的网络脚本会按照字母顺序处理配置文件。因此，您必须为配置文件命名，确保当依赖接口需要时，用于其他接口的规则和路由的接口会被启动。要实现正确的顺序，这个流程使用 `ifcfg-*`、`route-*` 和 `rules-*` 文件中的数字。

### 前提条件

- **NetworkManager** 软件包未安装，或者 **NetworkManager** 服务被禁用。
- **network-scripts** 软件包已安装。
- 要在流程中设置的 **RHEL** 路由器有四个网络接口：
  - **enp7s0** 接口已连接到提供商 A 的网络。提供商网络中的网关 IP 为 198.51.100.2，网络使用 /30 网络掩码。
  - **enp1s0** 接口已连接到提供商 B 的网络。提供商网络中的网关 IP 为 192.0.2.2，网络使用 /30 网络掩码。

- **enp8s0** 接口已与连有内部工作站的 10.0.0.0/24 子网相连。
- **enp9s0** 接口已与连有公司服务器的 203.0.113.0/24 子网相连。
- 内部工作站子网中的主机使用 10.0.0.1 作为默认网关。在此流程中，您可以将这个 IP 地址分配给路由器的 **enp8s0** 网络接口。
- 服务器子网中的主机使用 203.0.113.1 作为默认网关。在此流程中，您可以将这个 IP 地址分配给路由器的 **enp9s0** 网络接口。
- **firewalld** 服务已启用，并处于活动状态。

## 流程

1. 通过创建包含以下内容的 `/etc/sysconfig/network-scripts/ifcfg-1_Provider-A` 文件将网络接口的配置添加给提供商 A：

```
TYPE=Ethernet
IPADDR=198.51.100.1
PREFIX=30
GATEWAY=198.51.100.2
DNS1=198.51.100.200
DEFROUTE=yes
NAME=1_Provider-A
DEVICE=enp7s0
ONBOOT=yes
ZONE=external
```

配置文件使用以下参数：

- **TYPE=Ethernet**：定义连接类型为以太网。
- **IPADDR=*IP\_address***：设置 IPv4 地址。

- **PREFIX=*subnet\_mask*** : 设置子网掩码。
- **GATEWAY=*IP\_address*** : 设置默认网关地址。
- **DNS1=*IP\_of\_DNS\_server*** : 设置 DNS 服务器的 IPv4 地址。
- **DEFROUTE=*yes/no*** : 定义连接是否为默认路由。
- **NAME=*connection\_name*** : 设置连接配置文件的名称。使用有意义的名称以避免混淆。
- **DEVICE=*network\_device*** : 设置网络接口。
- **ONBOOT=*yes*** : 定义 RHEL 在系统引导时启动此连接。
- **ZONE=*firewalld\_zone*** : 将网络接口分配给定义的 firewalld 区域。请注意，firewalld 会为分配给 external 区域的接口自动启用伪装。

2. 为供应商 B 添加网络接口配置：

- a. 使用以下内容创建 /etc/sysconfig/network-scripts/ifcfg-2\_Provider-B 文件：

```
TYPE=Ethernet
IPADDR=192.0.2.1
PREFIX=30
DEFROUTE=no
NAME=2_Provider-B
DEVICE=enp1s0
ONBOOT=yes
ZONE=external
```

请注意，这个接口的配置文件不包含默认的网关设置。

b.

将 2\_Provider-B 连接的网关分配给单独的路由表。因此，使用以下内容创建 /etc/sysconfig/network-scripts/route-2\_Provider-B 文件：

```
0.0.0.0/0 via 192.0.2.2 table 5000
```

此条目将通过这个网关路由的所有子网的网关和流量分配给表 5000。

3.

为内部工作站子网创建网络接口配置：

a.

使用以下内容创建 /etc/sysconfig/network-scripts/ifcfg-3\_Internal-Workstations 文件：

```
TYPE=Ethernet
IPADDR=10.0.0.1
PREFIX=24
DEFROUTE=no
NAME=3_Internal-Workstations
DEVICE=enp8s0
ONBOOT=yes
ZONE=internal
```

b.

为内部工作站子网添加路由规则配置。因此，使用以下内容创建 /etc/sysconfig/network-scripts/rule-3\_Internal-Workstations 文件：

```
pri 5 from 10.0.0.0/24 table 5000
```

此配置定义了优先级为 5 的路由规则，该规则将来自 10.0.0.0/24 子网的所有流量路由到表 5000。低的值具有更高的优先级。

c.

使用以下内容创建 /etc/sysconfig/network-scripts/route-3\_Internal-Workstations 文件，以将静态路由添加到 ID 为 5000 的路由表：

```
10.0.0.0/24 via 192.0.2.1 table 5000
```

此静态路由定义 RHEL 将从 10.0.0.0/24 子网到本地网络接口的 IP 的流量发送给提供商 B (192.0.2.1)。这个接口是到路由表 5000，并用作下一跳。

4.

通过创建包含以下内容的 /etc/sysconfig/network-scripts/ifcfg-4\_Servers 文件来将网络接口的配置添加到服务器子网中：

```
TYPE=Ethernet
IPADDR=203.0.113.1
PREFIX=24
DEFROUTE=no
NAME=4_Servers
DEVICE=enp9s0
ONBOOT=yes
ZONE=internal
```

5.

重启网络：

```
systemctl restart network
```

## 验证

1.

在内部工作站子网的 RHEL 主机上：

a.

安装 traceroute 软件包：

```
yum install traceroute
```

b.

使用 traceroute 工具显示到互联网上主机的路由：

```
traceroute redhat.com
traceroute to redhat.com (209.132.183.105), 30 hops max, 60 byte packets
1 10.0.0.1 (10.0.0.1) 0.337 ms 0.260 ms 0.223 ms
2 192.0.2.2 (192.0.2.2) 0.884 ms 1.066 ms 1.248 ms
...
```

命令的输出显示路由器通过 192.0.2.1，即提供商 B 的网络来发送数据包。

2.

在服务器子网的 RHEL 主机上：

a.

安装 traceroute 软件包：

```
yum install traceroute
```

b.

使用 traceroute 工具显示到互联网上主机的路由：

```
traceroute redhat.com
traceroute to redhat.com (209.132.183.105), 30 hops max, 60 byte packets
 1 203.0.113.1 (203.0.113.1) 2.179 ms 2.073 ms 1.944 ms
 2 198.51.100.2 (198.51.100.2) 1.868 ms 1.798 ms 1.549 ms
...
```

命令的输出显示路由器通过 198.51.100.2，即供应商 A 的网络来发送数据包。

## 故障排除步骤

在 RHEL 路由器中：

1.

显示规则列表：

```
ip rule list
0: from all lookup local
5: from 10.0.0.0/24 lookup 5000
32766: from all lookup main
32767: from all lookup default
```

默认情况下，RHEL 包含表 local、main 和 default 的规则。

2.

显示表 5000 中的路由：

```
ip route list table 5000
default via 192.0.2.2 dev enp1s0
10.0.0.0/24 via 192.0.2.1 dev enp1s0
```

3.

显示接口和防火墙区：

```
firewall-cmd --get-active-zones
external
 interfaces: enp1s0 enp7s0
internal
 interfaces: enp8s0 enp9s0
```

4.

验证 **external** 区是否启用了伪装：

```
firewall-cmd --info-zone=external
external (active)
target: default
icmp-block-inversion: no
interfaces: enp1s0 enp7s0
sources:
services: ssh
ports:
protocols:
masquerade: yes
...
```

## 其他资源

- [使用旧网络脚本时，涉及基于策略的路由的配置文件概述](#)
- 您系统上的 **ip-route (8)** 和 **ip-rule (8)** 手册页
- [/usr/share/doc/network-scripts/sysconfig.txt](#) 文件

## 第 27 章 在不同的接口上重复使用相同的 IP 地址

使用虚拟路由和转发(VRF)，管理员可以在同一主机上同时使用多个路由表。为此，VRF 将网络在第 3 层进行分区。这可让管理员使用每个 VRF 域的独立路由表隔离流量。这个技术与虚拟 LAN（虚拟 LAN）类似，后者在第二层为网络分区，其中操作系统使用不同的 VLAN 标签来隔离共享相同物理介质的流量。

VRF 优于在第二层上分区的好处是，路由会根据涉及的对等者数量进行更好地考虑。

**Red Hat Enterprise Linux** 为每个 VRF 域使用虚拟 `virt` 设备，并通过向 VRF 设备添加现有网络设备来向 VRF 域添加路由。之前附加到原始设备的地址和路由将在 VRF 域中移动。

请注意，每个 VRF 域间都是相互隔离的。

### 27.1. 在不同接口上永久重复使用相同的 IP 地址

您可以使用虚拟路由和转发(VRF)功能来对一个服务器的不同接口永久使用同样的 IP 地址。



#### 重要

要在重新使用相同的 IP 地址时让远程对等两个 VRF 接口都联系，网络接口必须属于不同的广播域。网络中的广播域是一组节点，它们接收其中任何一个节点发送的广播流量。在大多数配置中，所有连接到同一交换机的节点都属于相同的域。

#### 先决条件

- 以 `root` 用户身份登录。
- 没有配置网络接口。

#### 流程

1. 创建并配置第一个 VRF 设备：
  - a.

为 VRF 设备创建连接并将其分配到路由表中。例如，要创建一个分配给 1001 路由表、名为 vrf0 的 VRF 设备：

```
nmcli connection add type vrf ifname vrf0 con-name vrf0 table 1001 ipv4.method disabled ipv6.method disabled
```

b.

启用 vrf0 设备：

```
nmcli connection up vrf0
```

c.

为刚刚创建的 VRF 分配网络设备。例如，要向 vrf0 VRF 设备添加 enp1s0 以太网设备，并向 enp1s0 分配 IP 地址和子网掩码，请输入：

```
nmcli connection add type ethernet con-name vrf.enp1s0 ifname enp1s0 master vrf0 ipv4.method manual ipv4.address 192.0.2.1/24
```

d.

激活 vrf.enp1s0 连接：

```
nmcli connection up vrf.enp1s0
```

2.

创建并配置下一个 VRF 设备：

a.

创建 VRF 设备并将其分配到路由表中。例如，要创建一个分配给 1002 路由表、名为 vrf1 的 VRF 设备，请输入：

```
nmcli connection add type vrf ifname vrf1 con-name vrf1 table 1002 ipv4.method disabled ipv6.method disabled
```

b.

激活 vrf1 设备：

```
nmcli connection up vrf1
```

c.

为刚刚创建的 VRF 分配网络设备。例如，要向 vrf1 VRF 设备添加 enp7s0 以太网设备，并给 enp7s0 分配 IP 地址和子网掩码，请输入：

```
nmcli connection add type ethernet con-name vrf.enp7s0 ifname enp7s0 master vrf1 ipv4.method manual ipv4.address 192.0.2.1/24
```

d.

激活 vrf.enp7s0 设备：

```
nmcli connection up vrf.enp7s0
```

## 27.2. 在不同接口中临时重复使用相同的 IP 地址

您可以使用虚拟路由和转发(VRF)功能来对一个服务器的不同接口临时使用同样的 IP 地址。这个过程仅用于测试目的，因为配置是临时的并在重启系统后会丢失。



### 重要

要在重新使用相同的 IP 地址时让远程对等两个 VRF 接口都联系，网络接口必须属于不同的广播域。广播域是一组节点，它们接收被其中任何一个发送的广播流量。在大多数配置中，所有连接到同一交换机的节点都属于相同的域。

### 先决条件

- 以 root 用户身份登录。
- 没有配置网络接口。

### 流程

1. 创建并配置第一个 VRF 设备：

a.

创建 VRF 设备并将其分配到路由表中。例如，要创建一个分配给 1001 路由表、名为 blue 的 VRF 设备：

```
ip link add dev blue type vrf table 1001
```

b.

启用 blue 设备：

```
ip link set dev blue up
```

c.

为 VRF 设备分配网络设备。例如，要向 blue VRF 设备添加 enp1s0 以太网设备：

```
ip link set dev enp1s0 master blue
```

d.

启用 enp1s0 设备：

```
ip link set dev enp1s0 up
```

e.

向 enp1s0 设备分配 IP 地址和子网掩码。例如，将其设为 192.0.2.1/24：

```
ip addr add dev enp1s0 192.0.2.1/24
```

2.

创建并配置下一个 VRF 设备：

a.

创建 VRF 设备并将其分配到路由表中。例如，要创建一个分配给 1002 路由表、名为 red 的 VRF 设备：

```
ip link add dev red type vrf table 1002
```

b.

启用 red 设备：

```
ip link set dev red up
```

c.

为 VRF 设备分配网络设备。例如，要向 red VRF 设备添加 enp7s0 以太网设备：

```
ip link set dev enp7s0 master red
```

d.

启用 enp7s0 设备：

```
ip link set dev enp7s0 up
```

e.

为 enp7s0 设备分配与 blue VRF 域中 enp1s0 设备所使用的相同的 IP 地址和子网掩码：

```
ip addr add dev enp7s0 192.0.2.1/24
```

3.

可选：创建更多 VRF 设备，如上所述。

### 27.3. 其他资源

●

`kernel-doc` 软件包的 `/usr/share/doc/kernel-doc-<kernel_version>/Documentation/networking/vrf.txt`

## 第 28 章 在隔离的 VRF 网络内启动服务

使用虚拟路由和转发(VRF)，您可以使用与操作系统主路由表不同的路由表创建隔离网络。然后，您可以启动服务和应用程序，以便它们只能访问该路由表中定义的网络。

### 28.1. 配置 VRF 设备

要使用虚拟路由和转发(VRF)，您可以创建一个 VRF 设备，并将物理或虚拟网络接口和路由信息附加给它。



#### 警告

要防止您将自己远程锁定，请在本地控制台中或通过您不想分配给 VRF 设备的网络接口远程执行此流程。

#### 先决条件

- 您已在本地登录或使用与您要分配给 VRF 设备不同的网络接口。

#### 流程

1.

使用同名的虚拟设备创建 vrf0 连接，并将其附加到路由表 1000：

```
nmcli connection add type vrf ifname vrf0 con-name vrf0 table 1000 ipv4.method disabled ipv6.method disabled
```

2.

向 vrf0 连接添加 enp1s0 设备，并配置 IP 设置：

```
nmcli connection add type ethernet con-name enp1s0 ifname enp1s0 master vrf0 ipv4.method manual/ipv4.address 192.0.2.1/24 ipv4.gateway 192.0.2.254
```

此命令会创建 enp1s0 连接，来作为 vrf0 连接的端口。由于此配置，路由信息会自动分配给与 vrf0 设备关联的路由表 1000。

3.

如果您在隔离网络中需要静态路由：

a.

添加静态路由：

```
nmcli connection modify enp1s0 +ipv4.routes "198.51.100.0/24 192.0.2.2"
```

这向 198.51.100.0/24 网络添加了一个路由，该网络使用 192.0.2.2 作为路由器。

b.

激活连接：

```
nmcli connection up enp1s0
```

验证

1.

显示与 vrf0 关联的设备的 IP 设置：

```
ip -br addr show vrf vrf0
enp1s0 UP 192.0.2.1/24
```

2.

显示 VRF 设备及其关联的路由表：

```
ip vrf show
Name Table

vrf0 1000
```

3.

显示主路由表：

```
ip route show
default via 203.0.113.0/24 dev enp7s0 proto static metric 100
```

主路由表不会提到任何与 enp1s0 设备或 192.0.2.1/24 子网关联的路由。

4.

显示路由表 1000：

```
ip route show table 1000
default via 192.0.2.254 dev enp1s0 proto static metric 101
broadcast 192.0.2.0 dev enp1s0 proto kernel scope link src 192.0.2.1
192.0.2.0/24 dev enp1s0 proto kernel scope link src 192.0.2.1 metric 101
local 192.0.2.1 dev enp1s0 proto kernel scope host src 192.0.2.1
broadcast 192.0.2.255 dev enp1s0 proto kernel scope link src 192.0.2.1
198.51.100.0/24 via 192.0.2.2 dev enp1s0 proto static metric 101
```

**default** 条目表示使用此路由表的服务，将 192.0.2.254 用作其默认网关，而不是主路由表中的默认网关。

5.

在与 vrf0 关联的网络中执行 traceroute 工具，以验证工具是否使用表 1000 的路由：

```
ip vrf exec vrf0 traceroute 203.0.113.1
traceroute to 203.0.113.1 (203.0.113.1), 30 hops max, 60 byte packets
1 192.0.2.254 (192.0.2.254) 0.516 ms 0.459 ms 0.430 ms
...
```

第一跳是分配给路由表 1000 的默认网关，而不是系统的主路由表中的默认网关。

## 其他资源



您系统上的 ip-vrf (8) 手册页

## 28.2. 在隔离的 VRF 网络内启动服务

您可以将服务（如 Apache HTTP 服务器）配置为在隔离的虚拟路由和转发(VRF)网络中启动。



**重要**

服务只能绑定到同一 VRF 网络中的本地 IP 地址。

## 先决条件



您已配置了 vrf0 设备。

- 您已将 Apache HTTP 服务器配置为仅侦听分配给与 vrf0 设备关联的接口的 IP 地址。

## 流程

- 显示 httpd systemd 服务的内容：

```
systemctl cat httpd
...
[Service]
ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND
...
```

在后续步骤中需要 ExecStart 参数的内容，以在隔离的 VRF 网络中运行相同的命令。

- 创建 /etc/systemd/system/httpd.service.d/ 目录：

```
mkdir /etc/systemd/system/httpd.service.d/
```

- 使用以下内容创建 /etc/systemd/system/httpd.service.d/override.conf 文件：

```
[Service]
ExecStart=
ExecStart=/usr/sbin/ip vrf exec vrf0 /usr/sbin/httpd $OPTIONS -DFOREGROUND
```

要覆盖 ExecStart 参数，您首先需要对其取消设置，然后将其设为所示的新值。

- 重新加载 systemd。

```
systemctl daemon-reload
```

- 重新启动 httpd 服务。

```
systemctl restart httpd
```

## 验证

1.

显示 httpd 进程的进程 ID(PID)：

```
pidof -c httpd
1904 ...
```

2.

显示 PID 的 VRF 关联，例如：

```
ip vrf identify 1904
vrf0
```

3.

显示与 vrf0 设备关联的所有 PID：

```
ip vrf pids vrf0
1904 httpd
...
...
```

## 其他资源



您系统上的 [ip-vrf \(8\)](#) 手册页

## 第 29 章 在 NETWORKMANAGER 连接配置文件中配置 ETHTOOL 设置

NetworkManager 可以永久配置某些网络驱动程序和硬件设置。与使用 ethtool 工具管理这些设置相比，这有重启后不会丢失设置的好处。

您可以在 NetworkManager 连接配置文件中设置以下 ethtool 设置：

### 卸载功能

网络接口控制器可以使用 TCP 卸载引擎(TOE)将某些操作卸载到网络接口控制器。这提高了网络吞吐量。

### 中断合并设置

通过使用中断合并，系统收集网络数据包，并为多个数据包生成一个中断。这会增加使用一个硬件中断发送到内核的数据量，从而减少中断负载，并最大化吞吐量。

### 环缓冲区

这些缓冲区存储传入和传出的网络数据包。您可以增加环缓冲区的大小，以减少高数据包丢弃率。

### 29.1. 使用 NMCLI 配置 ETHTOOL 卸载功能

您可以使用 NetworkManager 来在连接配置文件中启用和禁用 ethtool 卸载功能。

### 流程

1.

例如：要启用 RX 卸载特性，并在 `enp1s0` 连接配置文件中禁用 TX 卸载，请输入：

```
nmcli con modify enp1s0 ethtool.feature-rx on ethtool.feature-tx off
```

这个命令明确启用 RX 卸载并禁用 TX 卸载功能。

2.

要删除之前启用或禁用的卸载功能的设置，请将功能的参数设置为 `null` 值。例如，要删除 TX 卸载的配置，请输入：

```
nmcli con modify enp1s0 ethtool.feature-tx ""
```

3.

**重新激活网络配置集：**

```
nmcli connection up enp1s0
```

验证

•

**使用 ethtool -k 命令显示网络设备的当前卸载特性：**

```
ethtool -k network_device
```

其他资源

•

您系统上的 **nm-settings-nmcli (5)** 手册页

## 29.2. 使用 NETWORK RHEL 系统角色配置 ETHTOOL 卸载功能

网络接口控制器可以使用 TCP 卸载引擎(TOE)将某些操作卸载到网络接口控制器。这提高了网络吞吐量。您可以在网络接口的连接配置文件中配置卸载功能。通过使用 Ansible 和 **network RHEL** 系统角色，您可以自动化此过程，并在 **playbook** 中定义的主机上远程配置连接配置文件。



**警告**

您不能使用 **network RHEL** 系统角色只更新现有连接配置文件中的特定值。该角色确保连接配置文件与 **playbook** 中的设置完全匹配。如果具有相同名称的连接配置文件存在，则角色将应用 **playbook** 中的设置，并将配置文件中的所有其他设置重置为其默认值。要防止重置值，请始终在 **playbook** 中指定网络连接配置文件的整个配置，包括您不想更改的设置。

先决条件

•

您已准备好控制节点和受管节点

•

以可在受管主机上运行 **playbook** 的用户登录到控制节点。

- 您用于连接到受管节点的帐户对它们具有 **sudo** 权限。

## 流程

1.

创建一个包含以下内容的 **playbook** 文件，如 `~/playbook.yml`：

```

- name: Configure the network
 hosts: managed-node-01.example.com
 tasks:
 - name: Ethernet connection profile with dynamic IP address settings and offload
 features
 ansible.builtin.include_role:
 name: redhat.rhel_system_roles.network
 vars:
 network_connections:
 - name: enp1s0
 type: ethernet
 autoconnect: yes
 ip:
 dhcp4: yes
 auto6: yes
 ethtool:
 features:
 gro: no
 gso: yes
 tx_sctp_segmentation: no
 state: up
```

示例 **playbook** 中指定的设置包括如下：

**gro: no**

禁用通用接收卸载(GRO)。

**gso: yes**

启用通用分段卸载(GSO)。

**tx\_sctp\_segmentation: no**

禁用 TX 流控制传输协议(SCTP)分段。

有关 **playbook** 中使用的所有变量的详情，请查看控制节点上的

/usr/share/ansible/roles/rhel-system-roles.network/README.md 文件。

2.

验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不能防止错误的、但有效的配置。

3.

运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

●

查询受管节点的 Ansible 事实，并验证卸载设置：

```
ansible managed-node-01.example.com -m ansible.builtin.setup
...
 "ansible_enp1s0": {
 "active": true,
 "device": "enp1s0",
 "features": {
 ...
 "rx_gro_hw": "off",
 ...
 "tx_gso_list": "on",
 ...
 "tx_sctp_segmentation": "off",
 ...
 }
 ...
}
```

其他资源

●

/usr/share/ansible/roles/rhel-system-roles.network/README.md 文件

●

/usr/share/doc/rhel-system-roles/network/ 目录

## 29.3. 使用 NMCLI 配置 ETHTOOL COALESCE 设置

您可以使用 NetworkManager 来在连接配置文件中设置 ethtool 合并设置。

## 流程

1.

例如，要在 `enp1s0` 连接配置文件中将接收的数据包的最大数量设置为延迟到 128，请输入：

```
nmcli connection modify enp1s0 ethtool.coalesce-rx-frames 128
```

2.

要删除 coalesce 设置，请将其设置为 null 值。例如，要删除 `ethtool.coalesce-rx-frames` 设置，请输入：

```
nmcli connection modify enp1s0 ethtool.coalesce-rx-frames ""
```

3.

重新激活网络配置集：

```
nmcli connection up enp1s0
```

## 验证

- 

使用 `ethtool -c` 命令显示网络设备的当前卸载特性：

```
ethtool -c network_device
```

## 其他资源

- 

您系统上的 `nm-settings-nmcli (5)` 手册页

## 29.4. 使用 NETWORK RHEL 系统角色配置 ETHTOOL COALESCE 设置

通过使用中断合并，系统收集网络数据包，并为多个数据包生成一个中断。这会增加使用一个硬件中断发送到内核的数据量，从而减少中断负载，并最大化吞吐量。您可以在网络接口的连接配置文件中配置 coalesce 设置。通过使用 Ansible 和 network RHEL 角色，您可以自动化此过程，并在 playbook 中定义的主机上远程配置连接配置文件。



### 警告

您不能使用 **network RHEL** 系统角色只更新现有连接配置文件中的特定值。该角色确保连接配置文件与 **playbook** 中的设置完全匹配。如果具有相同名称的连接配置文件存在，则角色将应用 **playbook** 中的设置，并将配置文件中的所有其他设置重置为其默认值。要防止重置值，请始终在 **playbook** 中指定网络连接配置文件的整个配置，包括您不想更改的设置。

## 先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 **playbook** 的用户登录到控制节点。
- 您用于连接到受管节点的帐户对它们具有 **sudo** 权限。

## 流程

1.

创建一个包含以下内容的 **playbook** 文件，如 `~/playbook.yml`：

```

- name: Configure the network
 hosts: managed-node-01.example.com
 tasks:
 - name: Ethernet connection profile with dynamic IP address settings and coalesce
 settings
 ansible.builtin.include_role:
 name: redhat.rhel_system_roles.network
 vars:
 network_connections:
 - name: enp1s0
 type: ethernet
 autoconnect: yes
 ip:
 dhcp4: yes
 auto6: yes
 ethtool:
 coalesce:
```

```
rx_frames: 128
tx_frames: 128
state: up
```

示例 playbook 中指定的设置包括如下：

**rx\_frames: <value>**

设置 RX 帧的数量。

**gso: <value>**

设置 TX 帧的数量。

有关 playbook 中使用的所有变量的详情，请查看控制节点上的 /usr/share/ansible/roles/rhel-system-roles.network/README.md 文件。

2.

验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不能防止错误的、但有效的配置。

3.

运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

- 

显示网络设备的当前卸载功能：

```
ansible managed-node-01.example.com -m command -a 'ethtool -c enp1s0'
managed-node-01.example.com | CHANGED | rc=0 >>
...
rx-frames: 128
...
tx-frames: 128
...
```

## 其他资源

- `/usr/share/ansible/roles/rhel-system-roles.network/README.md` 文件
- `/usr/share/doc/rhel-system-roles/network/` 目录

### 29.5. 使用 NMCLI 增加环缓冲区的大小，以减少数据包丢弃率

如果数据包丢包率导致应用程序报告数据丢失、超时或其他问题，请增加以太网设备的环缓冲区的大小。

接收环缓冲在设备驱动程序和网络接口控制器(NIC)之间共享。该卡分配一个传输(TX)和接收(RX)环缓冲。名称意味着，环缓冲是循环缓冲区，溢出会覆盖现有数据。可以通过两种方法将数据从 NIC 移至内核，硬件中断和软件中断也称为 SoftIRQ。

内核使用 RX 环缓冲区来存储传入的数据包，直到设备驱动程序可以处理它们。设备驱动程序排空 RX 环，通常是使用 SoftIRQ，其将传入的数据包放在名为 `sk_buff` 或 `skb` 的内核数据结构中，以通过内核开始其过程，直到拥有相关套接字的应用程序。

内核使用 TX 环缓冲区来存放应发送到网络的传出数据包。这些环缓冲区位于堆栈的底部，是可能发生数据包丢弃的关键点，这反过来会对网络性能造成负面影响。

## 流程

1. 显示接口的数据包丢包统计信息：

```
ethtool -S enp1s0
...
rx_queue_0_drops: 97326
rx_queue_1_drops: 63783
...
```

请注意，命令的输出取决于网卡和驱动程序。

`discard` 或 `drop` 计数器中的高值表示可用缓冲区的填满速度快于内核可以处理数据包的速度。增加环缓冲有助于避免此类丢失。

2.

显示最大环缓冲大小：

```
ethtool -g enp1s0
Ring parameters for enp1s0:
Pre-set maximums:
RX: 4096
RX Mini: 0
RX Jumbo: 16320
TX: 4096
Current hardware settings:
RX: 255
RX Mini: 0
RX Jumbo: 0
TX: 255
```

如果 Pre-set maximums 部分中的值大于 Current hardware settings 部分的值，您可以在下一步中更改设置。

3.

确定使用接口的 NetworkManager 连接配置文件：

```
nmcli connection show
NAME UUID TYPE DEVICE
Example-Connection a5eb6490-cc20-3668-81f8-0314a27f3f75 ethernet enp1s0
```

4.

更新连接配置文件，并增加环缓冲：

- 要增加 RX 环缓冲区，请输入：

```
nmcli connection modify Example-Connection ethtool.ring-rx 4096
```

- 要增加 TX 环缓冲区，请输入：

```
nmcli connection modify Example-Connection ethtool.ring-tx 4096
```

5.

重新载入 NetworkManager 连接：

```
nmcli connection up Example-Connection
```

**重要**

根据 NIC 使用的驱动程序，环缓冲中的更改可能会短暂中断网络连接。

**其他资源**

- [ifconfig 和 ip 命令报告数据包丢弃（红帽知识库）](#)
- [我是否应该关注 0.05% 的数据包丢弃率？（红帽知识库）](#)
- 您系统上的 ethtool (8) 手册页

## 29.6. 使用 NETWORK RHEL 系统角色增加环缓冲区的大小，以减少高数据包丢弃率

如果数据包丢包率导致应用程序报告数据丢失、超时或其他问题，请增加以太网设备的环缓冲区的大小。

环缓冲区是循环缓冲区，溢出会覆盖现有数据。网卡分配一个传输(TX)和接收(RX)环缓冲区。接收环缓冲区在设备驱动程序和网络接口控制器(NIC)之间共享。数据可以通过硬件中断或软件中断（也称为 SoftIRQ）从 NIC 移到内核。

内核使用 RX 环缓冲区存储传入的数据包，直到设备驱动程序可以处理它们。设备驱动程序排空 RX 环，通常是使用 SoftIRQ，其将传入的数据包放在名为 `sk_buff` 或 `skb` 的内核数据结构中，以通过内核开始其过程，直到拥有相关套接字的应用程序。

内核使用 TX 环缓冲区来存放应发送到网络的传出数据包。这些环缓冲区位于堆栈的底部，是可能发生数据包丢弃的关键点，这反过来会对网络性能造成负面影响。

您可以在 NetworkManager 连接配置文件中配置环缓冲区设置。通过使用 Ansible 和 network RHEL 系统角色，您可以自动化此过程，并在 playbook 中定义的主机上远程配置连接配置文件。



### 警告

您不能使用 network RHEL 系统角色只更新现有连接配置文件中的特定值。该角色确保连接配置文件与 playbook 中的设置完全匹配。如果具有相同名称的连接配置文件存在，则角色将应用 playbook 中的设置，并将配置文件中的所有其他设置重置为其默认值。要防止重置值，请始终在 playbook 中指定网络连接配置文件的整个配置，包括您不想更改的设置。

## 先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接到受管节点的帐户具有 sudo 权限。
- 您知道设备支持的最大环缓冲区大小。

## 流程

1.

创建一个包含以下内容的 playbook 文件，如 ~/playbook.yml：

```

- name: Configure the network
 hosts: managed-node-01.example.com
 tasks:
 - name: Ethernet connection profile with dynamic IP address setting and increased
 ring buffer sizes
 ansible.builtin.include_role:
 name: redhat.rhel_system_roles.network
 vars:
 network_connections:
 - name: enp1s0
 type: ethernet
 autoconnect: yes
 ip:
 dhcp4: yes
 auto6: yes
```

```
ethtool:
ring:
 rx: 4096
 tx: 4096
state: up
```

示例 playbook 中指定的设置包括以下内容：

**rx: <value>**

设置接收的环缓冲区条目的最大数量。

**tx: <value>**

设置传输的环缓冲区条目的最大数量。

有关 playbook 中使用的所有变量的详情，请查看控制节点上的 /usr/share/ansible/roles/rhel-system-roles.network/README.md 文件。

2.

验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不会防止错误但有效的配置。

3.

运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

●

显示最大环缓冲大小：

```
ansible managed-node-01.example.com -m command -a 'ethtool -g enp1s0'
managed-node-01.example.com | CHANGED | rc=0 >>
...
Current hardware settings:
RX: 4096
```

```
RX Mini: 0
RX Jumbo: 0
TX: 4096
```

## 其他资源

- [`/usr/share/ansible/roles/rhel-system-roles.network/README.md` 文件](#)
- [`/usr/share/doc/rhel-system-roles/network/` 目录](#)

## 第 30 章 网络管理器调试介绍

对所有或某些域增加日志级别有助于记录 NetworkManager 所执行的操作的更多详情。您可以使用这些信息排查问题。NetworkManager 提供不同的级别和域来生成日志信息。/etc/NetworkManager/NetworkManager.conf 文件是 NetworkManager 的主配置文件。日志存储在日志中。

### 30.1. NETWORKMANAGER 重新应用方法的简介

NetworkManager 服务使用配置文件管理设备的连接设置。桌面总线(D-Bus) API 可以创建、修改和删除这些连接设置。对于配置文件中的任何更改，D-Bus API 会将现有的设置克隆到连接的修改设置中。虽然克隆，但更改不会应用到修改后的设置。要使它生效，请重新激活连接的现有设置或使用 `reapply()` 方法。

`reapply()` 方法具有以下特性：

1. 在不停用或重启网络接口的情况下更新修改后的连接设置。
2. 从修改后的连接设置中删除待处理的更改。因为 NetworkManager 不会恢复手动更改，因此您可以重新配置设备，并恢复外部或手动参数。
3. 创建与现有连接设置不同的修改后的连接设置。

另外，`reapply()` 方法支持以下属性：

- `bridge.ageing-time`
- `bridge.forward-delay`
- `bridge.group-address`

- **bridge.group-forward-mask**
- **bridge.hello-time**
- **bridge.max-age**
- **bridge.multicast-hash-max**
- **bridge.multicast-last-member-count**
- **bridge.multicast-last-member-interval**
- **bridge.multicast-membership-interval**
- **bridge.multicast-querier**
- **bridge.multicast-querier-interval**
- **bridge.multicast-query-interval**
- **bridge.multicast-query-response-interval**
- **bridge.multicast-query-use-ifaddr**
- **bridge.multicast-router**

- **bridge.multicast-snooping**
- **bridge.multicast-startup-query-count**
- **bridge.multicast-startup-query-interval**
- **bridge.priority**
- **bridge.stp**
- **bridge.VLAN-filtering**
- **bridge.VLAN-protocol**
- **bridge.VLANs**
- **802-3-ethernet.accept-all-mac-addresses**
- **802-3-ethernet.cloned-mac-address**
- **IPv4.addresses**
- **IPv4.dhcp-client-id**
- **IPv4.dhcp-iaid**

- **IPv4.dhcp-timeout**
- **IPv4.DNS**
- **IPv4.DNS-priority**
- **IPv4.DNS-search**
- **IPv4.gateway**
- **IPv4.ignore-auto-DNS**
- **IPv4.ignore-auto-routes**
- **IPv4.may-fail**
- **IPv4.method**
- **IPv4.never-default**
- **IPv4.route-table**
- **IPv4.routes**
- **IPv4.routing-rules**

- **IPv6.addr-gen-mode**
- **IPv6.addresses**
- **IPv6.dhcp-duid**
- **IPv6.dhcp-iaid**
- **IPv6.dhcp-timeout**
- **IPv6.DNS**
- **IPv6.DNS-priority**
- **IPv6.DNS-search**
- **IPv6.gateway**
- **IPv6.ignore-auto-DNS**
- **IPv6.may-fail**
- **IPv6.method**
- **IPv6.never-default**

- IPv6.ra-timeout
- IPv6.route-metric
- IPv6.route-table
- IPv6.routes
- IPv6.routing-rules

## 其他资源

- 您系统上的 **nm-settings-nmcli (5)** 手册页

### 30.2. 设置 NETWORKMANAGER 日志级别

默认情况下，所有日志域都设为记录 **INFO** 日志级别。在收集调试日志前禁用速率限制。通过速率限制，如果短时间内有太多信息，**systemd-journald** 会丢弃它们。当日志级别为 **TRACE** 时，可能会发生这种情况。

此流程禁用速率限制，并为所有（**ALL**）域启用记录调试日志。

## 流程

1. 要禁用速率限制，请编辑 **/etc/systemd/journald.conf** 文件，取消 [**Journal**] 部分中的 **RateLimitBurst** 参数的注释，并将其值设为 **0**：

```
RateLimitBurst=0
```

2. 重启 **systemd-journald** 服务。

```
systemctl restart systemd-journald
```

3.

使用以下内容创建 /etc/NetworkManager/conf.d/95-nm-debug.conf 文件：

```
[logging]
domains=ALL:TRACE
```

**domains** 参数可以包含多个用逗号分隔的 **domain:level** 对。

4.

重启 NetworkManager 服务。

```
systemctl restart NetworkManager
```

验证



查询 **systemd** 日志以显示 NetworkManager 单元的日志条目：

```
journalctl -u NetworkManager
...
Jun 30 15:24:32 server NetworkManager[164187]: <debug> [1656595472.4939] active-connection[0x5565143c80a0]: update activation type from assume to managed
Jun 30 15:24:32 server NetworkManager[164187]: <trace> [1656595472.4939]
device[55b33c3bdb72840c] (enp1s0): sys-iface-state: assume -> managed
Jun 30 15:24:32 server NetworkManager[164187]: <trace> [1656595472.4939]
l3cfg[4281fdf43e356454,ifindex=3]: commit type register (type "update", source
"device", existing a369f23014b9ede3) -> a369f23014b9ede3
Jun 30 15:24:32 server NetworkManager[164187]: <info> [1656595472.4940] manager:
NetworkManager state is now CONNECTED_SITE
...
...
```

### 30.3. 使用 NMCLI 临时设置运行时的日志级别

您可以使用 **nmcli** 更改运行时的日志级别。

流程

1.

可选：显示当前的日志设置：

```
nmcli general logging
LEVEL DOMAINS
INFO
PLATFORM,RFKILL,ETHER,WIFI,BT,MB,DHCP4,DHCP6,PPP,WIFI_SCAN,IP4,IP6,A
```

**UTOIP4,DNS,VPN,SHARING,SUPPLICANT,AGENTS,SETTINGS,SUSPEND,CORE,DEVICE,OLPC,WIMAX,INFINIBAND,FIREWALL,ADSL,BOND,VLAN,BRIDGE,DBUS\_PROPS,TEAM,CON\_CHECK,DCB,DISPATCH**

2.

要修改日志级别和域，请使用以下选项：

- 要将所有域的日志级别都设为同样的 **LEVEL**，请输入：

**# nmcli general logging level *LEVEL* domains *ALL***

- 要更改特定域的级别，请输入：

**# nmcli general logging level *LEVEL* domains *DOMAINS***

请注意，使用这个命令更新日志级别会禁用所有其他域的日志功能。

- 要更改特定域的级别并保持其它域的级别，请输入：

**# nmcli general logging level KEEP domains *DOMAIN:LEVEL,DOMAIN:LEVEL***

#### 30.4. 查看 NETWORKMANAGER 日志

您可以查看 NetworkManager 日志进行故障排除。

##### 流程

- 要查看日志，请输入：

**# journalctl -u NetworkManager -b**

##### 其他资源

- 您系统上的 NetworkManager.conf (5) 和 journalctl (1) 手册页

### 30.5. 调试级别和域

您可以使用 **levels** 和 **domains** 参数来管理 NetworkManager 的调试。**levels** 定义了详细程度，而 **domains** 定义了消息的类别，以记录给定的严重程度（级别）的日志。

日志级别	描述
<b>OFF</b>	不记录任何有关 NetworkManager 的信息
<b>ERR</b>	仅记录严重错误
<b>WARN</b>	记录可以反映操作的警告信息
<b>INFO</b>	记录各种有助于跟踪状态和操作的信息
<b>DEBUG</b>	为调试启用详细日志记录
<b>TRACE</b>	启用比 <b>DEBUG</b> 级别更详细的日志

请注意，后续的级别记录来自以前级别的所有信息。例如，将日志级别设为 **INFO** 也会记录包含在 **ERR** 和 **WARN** 日志级别中的消息。

#### 其他资源

- 您系统上的 **NetworkManager.conf (5)** 手册页

## 第 31 章 使用 LLDP 来调试网络配置问题

您可以使用链路层发现协议(LLDP)来调试拓扑中的网络配置问题。这意味着 LLDP 可以报告与其他主机或路由器以及交换机的配置不一致。

### 31.1. 使用 LLDP 信息调试不正确的 VLAN 配置

如果您将交换机端口配置为使用特定的 VLAN，而主机没有收到这些 VLAN 数据包，则您可以使用链路层发现协议(LLDP)来调试问题。在没有收到数据包的主机上执行这个流程。

#### 先决条件

- nmstate 软件包已安装。
- 交换机支持 LLDP。
- LLDP 在邻居设备上已启用。

#### 流程

1. 使用以下内容创建 ~/enable-LDP-enp1s0.yml 文件：
 

```
interfaces:
 - name: enp1s0
 type: ethernet
 lldp:
 enabled: true
```
2. 使用 ~/enable-LDP-enp1s0.yml 文件来在接口 enp1s0 上启用 LLDP：
 

```
nmstatectl apply ~/enable-LDP-enp1s0.yml
```
3. 显示 LLDP 信息：
 

```
nmstatectl show enp1s0
 - name: enp1s0
```

```
type: ethernet
state: up
ipv4:
 enabled: false
 dhcp: false
ipv6:
 enabled: false
 autoconf: false
 dhcp: false
lldp:
 enabled: true
 neighbors:
 - type: 5
 system-name: Summit300-48
 - type: 6
 system-description: Summit300-48 - Version 7.4e.1 (Build 5)
 05/27/05 04:53:11
 - type: 7
 system-capabilities:
 - MAC Bridge component
 - Router
 - type: 1
 _description: MAC address
 chassis-id: 00:01:30:F9:AD:A0
 chassis-id-type: 4
 - type: 2
 _description: Interface name
 port-id: 1/1
 port-id-type: 5
 - type: 127
 ieee-802-1-vlans:
 - name: v2-0488-03-0505
 vid: 488
 oui: 00:80:c2
 subtype: 3
 - type: 127
 ieee-802-3-mac-phy-conf:
 autoneg: true
 operational-mau-type: 16
 pmd-autoneg-cap: 27648
 oui: 00:12:0f
 subtype: 1
 - type: 127
 ieee-802-1-ppvids:
 - 0
 oui: 00:80:c2
 subtype: 2
 - type: 8
 management-addresses:
 - address: 00:01:30:F9:AD:A0
 address-subtype: MAC
 interface-number: 1001
 interface-number-subtype: 2
 - type: 127
 ieee-802-3-max-frame-size: 1522
 oui: 00:12:0f
```

```
subtype: 4
mac-address: 82:75:BE:6F:8C:7A
mtu: 1500
```

4.

验证输出，以确保设置与您预期的配置匹配。例如，连接到交换机的接口的 LLDP 信息显示此主机连接的交换机端口使用 VLAN ID 448：

```
- type: 127
ieee-802-1-vlans:
- name: v2-0488-03-0505
 vid: 488
```

如果 `enp1s0` 接口的网络配置使用不同的 VLAN ID，请相应地进行修改。

## 其他资源

[配置 VLAN 标记](#)

## 第 32 章 LINUX 流量控制

Linux 提供管理和操作数据包传输的工具。Linux 流量控制 (TC) 子系统帮助进行策略、分类、控制以及调度网络流量。TC 还可以通过使用过滤器和动作在分类过程中利用数据包内容分栏。TC 子系统通过使用排队规则(**qdisc**)这个 TC 架构的基本元素来达到此目的。

调度机制在进入或退出不同的队列前确定或者重新安排数据包。最常见的调度程序是先入先出 (FIFO) 调度程序。您可以使用 **tc** 工具临时执行 **qdiscs** 操作，也可以使用 **NetworkManager** 永久执行操作。

在 Red Hat Enterprise Linux 中，您可以使用各种方法配置默认的排队规则来管理网络接口上的流量。

### 32.1. 排队规则概述

排队规则(**qdiscs**)帮助排队，之后通过网络接口调度流量传输。**qdisc** 有两个操作：

- 排队请求，以便在以后传输时对数据包进行排队
- 出队请求，以便可以选择其中一个排队的数据包进行即时传输。

每个 **qdisc** 都有一个 16 位十六进制标识数字，称为 句柄，带有一个附加的冒号，如 1: 或 abcd : 这个数字被称为 **qdisc** 主号码。如果 **qdisc** 有类，则标识符是由两个数字组成的对，主号码在次要号码之前，即 **<major>:<minor>**，例如 abcd:1。次要号码的编号方案取决于 **qdisc** 类型。有时，编号是系统化的，其中第一类的 ID 为 **<>:1**，第二类的 ID 为 **<>:2**，等等。一些 **qdiscs** 允许用户在创建类时随机设置类的次要号码。

#### 类 **qdiscs**

存在不同类型的 **qdiscs**，帮助向和从网络接口传输数据包。您可以使用根、父或子类配置 **qdiscs**。子对象可以被附加的位置称为类。**qdisc** 中的类是灵活的，始终包含多个子类或一个子类 **qdisc**。不禁止包含类 **qdisc** 本身的类，这有助于实现复杂的流量控制场景。

类 **qdiscs** 本身不存储任何数据包。相反，它们根据特定于 **qdisc** 的标准，将排队和出队请求传到它们其中的一个子类。最后，这个递归数据包传递最终结束保存数据包的位置（在出现排队时从中提取）。

#### 无类别 **qdiscs**

有些 qdiscs 不包含子类，它们称为无类别 qdiscs。与类 qdiscs 相比，无类别 qdiscs 需要较少的自定义。通常情况下，将它们附加到接口就足够了。

## 其他资源

- 您系统上的 tc (8) 和 tc-actions (8) 手册页

### 32.2. 使用 TC 工具检查网络接口的 QDISCS

默认情况下，Red Hat Enterprise Linux 系统使用 fq\_codel qdisc。您可以使用 tc 工具检查 qdisc 计数器。

## 流程

1. 可选：查看您当前的 qdisc：

```
tc qdisc show dev enp0s1
```

2. 检查当前的 qdisc 计数器：

```
tc -s qdisc show dev enp0s1
qdisc fq_codel 0: root refcnt 2 limit 10240p flows 1024 quantum 1514 target 5.0ms
interval 100.0ms memory_limit 32Mb ecn
Sent 1008193 bytes 5559 pkt (dropped 233, overlimits 55 requeues 77)
backlog 0b 0p requeues 0
```

- **dropped** - 由于所有队列已满而丢弃数据包的次数
- **overlimits** - 配置的链路容量已满的次数
- **sent** - 出队的数量

### 32.3. 更新默认的 QDISC

如果使用当前的 qdisc 观察网络数据包丢失情况，您可以根据您的网络要求更改 qdisc。

## 流程

1.

查看当前的默认的 qdisc：

```
sysctl -a | grep qdisc
net.core.default_qdisc = fq_codel
```

2.

查看当前以太网连接的 qdisc：

```
tc -s qdisc show dev enp0s1
qdisc fq_codel 0: root refcnt 2 limit 10240p flows 1024 quantum 1514 target 5.0ms
interval 100.0ms memory_limit 32Mb ecn
Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
backlog 0b 0p requeues 0
maxpacket 0 drop_overlimit 0 new_flow_count 0 ecn_mark 0
new_flows_len 0 old_flows_len 0
```

3.

更新现有的 qdisc：

```
sysctl -w net.core.default_qdisc=pfifo_fast
```

4.

要应用更改，请重新加载网络驱动程序：

```
modprobe -r NETWORKDRIVERNAME
modprobe NETWORKDRIVERNAME
```

5.

启动网络接口：

```
ip link set enp0s1 up
```

## 验证

•

查看以太网连接的 qdisc：

```
tc -s qdisc show dev enp0s1
qdisc pfifo_fast 0: root refcnt 2 bands 3 priomap 1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
```

```
Sent 373186 bytes 5333 pkt (dropped 0, overlimits 0 requeues 0)
backlog 0b 0p requeues 0
....
```

## 其他资源

- [如何在 Red Hat Enterprise Linux 上设置 sysctl 变量 \(红帽知识库\)](#)

## 32.4. 使用 TC 工具临时设置网络接口的当前 QDISC

您可以更新当前的 qdisc 而不更改默认的 qdisc。

### 流程

1. 可选：查看当前的 qdisc：

```
tc -s qdisc show dev enp0s1
```

2. 更新当前的 qdisc：

```
tc qdisc replace dev enp0s1 root htb
```

### 验证

- 查看更新后的当前 qdisc：

```
tc -s qdisc show dev enp0s1
qdisc htb 8001: root refcnt 2 r2q 10 default 0 direct_packets_stat 0 direct_qlen 1000
Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
backlog 0b 0p requeues 0
```

## 32.5. 使用 NETWORKMANAGER 永久设置当前网络接口的 QDISC

您可以更新 NetworkManager 连接当前的 qdisc 值。

### 流程

1.

可选：查看当前的 qdisc：

```
tc qdisc show dev enp0s1
qdisc fq_codel 0: root refcnt 2
```

2.

更新当前的 qdisc：

```
nmcli connection modify enp0s1 tc.qdiscs 'root pfifo_fast'
```

3.

可选：要在现有的 qdisc 上添加另一个 qdisc，请使用 `+tc.qdisc` 选项：

```
nmcli connection modify enp0s1 +tc.qdisc 'ingress handle ffff:'
```

4.

激活更改：

```
nmcli connection up enp0s1
```

## 验证

●

查看网络接口当前的 qdisc：

```
tc qdisc show dev enp0s1
qdisc pfifo_fast 8001: root refcnt 2 bands 3 priomap 1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
qdisc ingress ffff: parent ffff:ffff1 -----
```

## 其他资源

●

您系统上的 `nm-settings (5)` 手册页

## 32.6. RHEL 中可用的 QDISCS

每个 qdisc 解决唯一的与网络相关的问题。以下是 RHEL 中可用的 qdiscs 列表。您可以使用以下任何一个 qdisc 来根据您的网络要求来塑造网络流量。

表 32.1. RHEL 中的可用调度程序

qdisc 名称	包含在	卸载支持
异步传输模式(ATM)	<b>kernel-modules-extra</b>	
基于信用的塑造程序	<b>kernel-modules-extra</b>	是
CHOOSE 和 Keep 用于有响应的流量，CHOOSE 和 Kill 用于没有响应的流量 (CHOKE)	<b>kernel-modules-extra</b>	
受控的延迟 (CoDel)	<b>kernel-core</b>	
Enhanced Transmission Selection (ETS)	<b>kernel-modules-extra</b>	是
Fair Queue (FQ)	<b>kernel-core</b>	
Fair Queuing Controlled Delay (FQ_CoDel)	<b>kernel-core</b>	
Generalized Random Early Detection (GRED)	<b>kernel-modules-extra</b>	
Hierarchical Fair Service Curve (HSFC)	<b>kernel-core</b>	
Heavy-Hitter Filter (HHF)	<b>kernel-core</b>	
Hierarchy Token Bucket (HTB)	<b>kernel-core</b>	是
INGRESS	<b>kernel-core</b>	是
Multi Queue Priority (MQPRIO)	<b>kernel-modules-extra</b>	是
Multiqueue (MULTIQ)	<b>kernel-modules-extra</b>	是
Network Emulator (NETEM)	<b>kernel-modules-extra</b>	
Proportional Integral-controller Enhanced (PIE)	<b>kernel-core</b>	
PLUG	<b>kernel-core</b>	
Quick Fair Queueing (QFQ)	<b>kernel-modules-extra</b>	
Random Early Detection (RED)	<b>kernel-modules-extra</b>	是

qdisc 名称	包含在	卸载支持
Stochastic Fair Blue (SFB)	<b>kernel-modules-extra</b>	
Stochastic Fairness Queueing (SFQ)	<b>kernel-core</b>	
Token Bucket Filter (TBF)	<b>kernel-core</b>	是
Trivial Link Equalizer (TEQL)	<b>kernel-modules-extra</b>	

**重要**

**qdisc 卸载需要对 NIC 的硬件和驱动程序的支持。**

## 其他资源

- 您系统上的 **tc (8)** 手册页

## 第 33 章 使用 802.1X 标准和存储在文件系统上的证书对 RHEL 客户端进行网络验证

管理员通常使用基于 IEEE 802.1X 标准的基于端口的网络访问控制（NAC）来保护网络不受未授权 LAN 和 Wi-Fi 客户端的影响。如果网络使用可扩展身份验证协议传输层安全(EAP-TLS)机制，则您需要证书来向这个网络验证客户端。

### 33.1. 使用 NMCLI 在现有以太网连接上配置 802.1X 网络身份验证

您可以使用 nmcli 工具在命令行上配置带有 802.1X 网络身份验证的以太网连接。

#### 先决条件

- 网络支持 802.1X 网络身份验证。
- 以太网连接配置集存在于 NetworkManager 中，且具有有效的 IP 配置。
- 客户端上存在 TLS 身份验证所需的以下文件：
  - 存储的客户端密钥位于 /etc/pki/tls/private/client.key 文件中，该文件归 root 用户所有，且只对 root 可读。
  - 客户端证书存储在 /etc/pki/tls/certs/client.crt 文件中。
  - 证书颁发机构(CA)证书存储在 /etc/pki/tls/certs/ca.crt 文件中。
- wpa\_supplicant 软件包已安装。

#### 流程

1. 将扩展验证协议(EAP)设置为 tls，将路径设置为客户端证书和密钥文件：

```
nmcli connection modify enp1s0 802-1x.eap tls 802-1x.client-cert
/etc/pki/tls/certs/client.crt 802-1x.private-key /etc/pki/tls/certs/certs/client.key
```

请注意，您必须在一个命令中设置 `802-1x.eap`、`802-1x.client-cert` 和 `802-1x.private-key` 参数。

2.

设置 CA 证书的路径：

```
nmcli connection modify enp1s0 802-1x.ca-cert /etc/pki/tls/certs/ca.crt
```

3.

设置证书中使用的用户的身份：

```
nmcli connection modify enp1s0 802-1x.identity user@example.com
```

4.

可选：将密码存储在配置中：

```
nmcli connection modify enp1s0 802-1x.private-key-password password
```

### 重要

默认情况下，NetworkManager 将密码以明文形式存储在磁盘上的连接配置文件中，但该文件只对 root 用户可读。但是，在配置文件中清除文本密码会有安全隐患。

要提高安全性，请将 `802-1x.password-flags` 参数设置为 `agent-owned`。使用这个设置，在具有 GNOME 桌面环境或运行 `nm-applet` 的服务器上，NetworkManager 在解锁密钥环后从这些服务检索密码。在其他情况下，NetworkManager 会提示输入密码。

5.

激活连接配置集：

```
nmcli connection up enp1s0
```

### 验证

- 访问需要网络身份验证的网络上的资源。

## 其他资源

- [配置以太网连接](#)

### 33.2. 使用 NMSTATECTL 配置带有 802.1X 网络身份验证的静态以太网连接

使用 `nmstatectl` 工具，通过 `Nmstate API` 配置带有 802.1X 网络身份验证的以太网连接。`Nmstate API` 确保设置配置后结果与配置文件匹配。如果有任何失败，`nmstatectl` 会自动回滚更改以避免系统处于不正确的状态。



#### 注意

`nmstate` 库只支持 TLS 可扩展身份验证协议(EAP)方法。

## 先决条件

- 网络支持 802.1X 网络身份验证。
- 受管节点使用 `NetworkManager`。
- 客户端上存在 TLS 身份验证所需的以下文件：
  - 存储的客户端密钥位于 `/etc/pki/tls/private/client.key` 文件中，该文件归 `root` 用户所有，且只对 `root` 可读。
  - 客户端证书存储在 `/etc/pki/tls/certs/client.crt` 文件中。
  - 证书颁发机构(CA)证书存储在 `/etc/pki/tls/certs/ca.crt` 文件中。

## 流程

1.

创建包含以下内容的 YAML 文件，如 `~/create-ethernet-profile.yml`：

```

interfaces:
- name: enp1s0
 type: ethernet
 state: up
 ipv4:
 enabled: true
 address:
 - ip: 192.0.2.1
 prefix-length: 24
 dhcp: false
 ipv6:
 enabled: true
 address:
 - ip: 2001:db8:1::1
 prefix-length: 64
 autoconf: false
 dhcp: false
 802.1x:
 ca-cert: /etc/pki/tls/certs/ca.crt
 client-cert: /etc/pki/tls/certs/client.crt
 eap-methods:
 - tls
 identity: client.example.org
 private-key: /etc/pki/tls/private/client.key
 private-key-password: password
 routes:
 config:
 - destination: 0.0.0.0/0
 next-hop-address: 192.0.2.254
 next-hop-interface: enp1s0
 - destination: ::/0
 next-hop-address: 2001:db8:1::fffe
 next-hop-interface: enp1s0
 dns-resolver:
 config:
 search:
 - example.com
 server:
 - 192.0.2.200
 - 2001:db8:1::ffbb
```

这些设置使用以下设置为 `enp1s0` 设备定义一个以太网连接配置文件：



- 静态 IPv4 地址 - 192.0.2.1 和 /24 子网掩码



- 静态 IPv6 地址 - 2001:db8:1::1 和 /64 子网掩码
  - IPv4 默认网关 - 192.0.2.254
  - IPv6 默认网关 - 2001:db8:1::ffffe
  - IPv4 DNS 服务器 - 192.0.2.200
  - IPv6 DNS 服务器 - 2001:db8:1::ffbb
  - DNS 搜索域 - example.com
  - 使用 TLS EAP 协议的 802.1x 网络身份验证
2. 将设置应用到系统：

```
nmstatectl apply ~/create-ethernet-profile.yml
```

验证

- 访问需要网络身份验证的网络上的资源。

### 33.3. 使用 NETWORK RHEL 系统角色配置具有 802.1X 网络身份验证的静态以太网连接

网络访问控制(NAC)保护网络免受未授权客户端访问。您可以在 NetworkManager 连接配置文件中指定身份验证所需的详情，以使客户端可以访问网络。通过使用 Ansible 和 network RHEL 系统角色，您可以自动化此过程，并在 playbook 中定义的主机上远程配置连接配置文件。

您可以使用 Ansible playbook 将私钥、证书和 CA 证书复制到客户端，然后使用 network RHEL 系统角色配置具有 802.1X 网络身份验证的连接配置文件。

## 先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 您用于连接到受管节点的帐户对它们具有 sudo 权限。
- 网络支持 802.1X 网络身份验证。
- 受管节点使用 NetworkManager。
- 以下 TLS 身份验证所需的文件在控制节点上存在：
  - 客户端密钥存储在 /srv/data/client.key 文件中。
  - 客户端证书存储在 /srv/data/client.crt 文件中。
  - 证书颁发机构(CA)证书存储在 /srv/data/ca.crt 文件中。

## 流程

1. 将敏感变量存储在加密的文件中：
  - a. 创建 vault：

```
$ ansible-vault create ~/vault.yml
New Vault password: <vault_password>
Confirm New Vault password: <vault_password>
```

b.

在 ansible-vault create 命令打开编辑器后，以 `<key>: <value>` 格式输入敏感数据：

```
pwd: <password>
```

c.

保存更改，并关闭编辑器。Ansible 加密 vault 中的数据。

2.

创建一个包含以下内容的 playbook 文件，如 `~/playbook.yml`：

```

- name: Configure an Ethernet connection with 802.1X authentication
 hosts: managed-node-01.example.com
 vars_files:
 - ~/vault.yml
 tasks:
 - name: Copy client key for 802.1X authentication
 ansible.builtin.copy:
 src: "/srv/data/client.key"
 dest: "/etc/pki/tls/private/client.key"
 mode: 0600

 - name: Copy client certificate for 802.1X authentication
 ansible.builtin.copy:
 src: "/srv/data/client.crt"
 dest: "/etc/pki/tls/certs/client.crt"

 - name: Copy CA certificate for 802.1X authentication
 ansible.builtin.copy:
 src: "/srv/data/ca.crt"
 dest: "/etc/pki/ca-trust/source/anchors/ca.crt"

 - name: Ethernet connection profile with static IP address settings and 802.1X
 ansible.builtin.include_role:
 name: redhat.rhel_system_roles.network
 vars:
 network_connections:
 - name: enp1s0
 type: ethernet
 autoconnect: yes
 ip:
 address:
 - 192.0.2.1/24
 - 2001:db8:1::1/64
 gateway4: 192.0.2.254
 gateway6: 2001:db8:1::fffe
 dns:
 - 192.0.2.200
 - 2001:db8:1::ffbb
 dns_search:
 - example.com
```

```
ieee802_1x:
 identity: <user_name>
 eap: tls
 private_key: "/etc/pki/tls/private/client.key"
 private_key_password: "{{ pwd }}"
 client_cert: "/etc/pki/tls/certs/client.crt"
 ca_cert: "/etc/pki/ca-trust/source/anchors/ca.crt"
 domain_suffix_match: example.com
 state: up
```

示例 playbook 中指定的设置包括如下：

### ieee802\_1x

此变量包含与 802.1X 相关的设置。

#### eap: tls

将配置文件配置为对可扩展身份验证协议(EAP)使用基于证书的 TLS 身份验证方法。

有关 playbook 中使用的所有变量的详情，请查看控制节点上的 /usr/share/ansible/roles/rhel-system-roles.network/README.md 文件。

3.

验证 playbook 语法：

```
$ ansible-playbook --ask-vault-pass --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不能防止错误的、但有效的配置。

4.

运行 playbook：

```
$ ansible-playbook --ask-vault-pass ~/playbook.yml
```

验证

●

访问需要网络身份验证的网络上的资源。

其他资源

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** 文件
- **/usr/share/doc/rhel-system-roles/network/** directory
- **Ansible vault**

## 第 34 章 使用带有 FREERADIUS 后端的 HOSTAPD 为 LAN 客户端设置 802.1X 网络身份验证服务

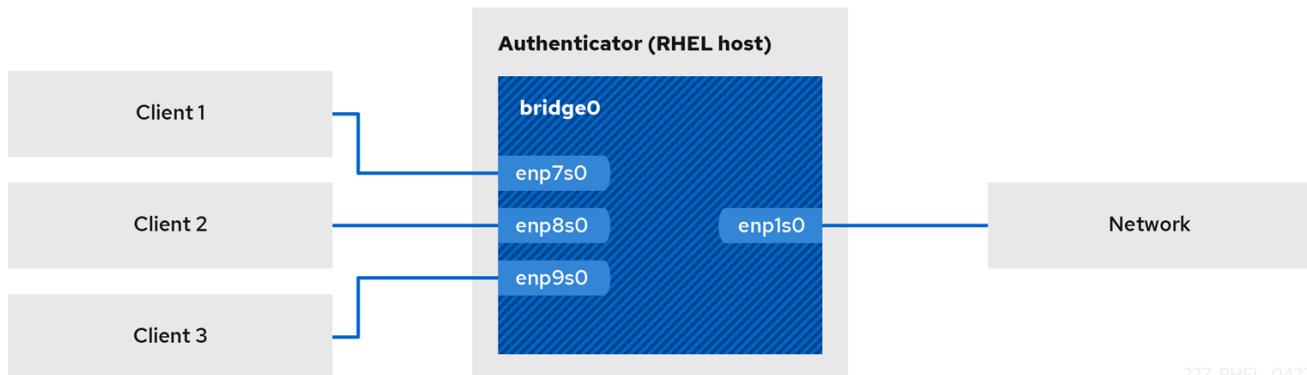
IEEE 802.1X 标准定义了安全身份验证和授权方法，以保护网络不接受未授权的客户端。通过使用 hostapd 服务和 FreeRADIUS，您可以在您的网络中提供网络访问控制(NAC)。



### 注意

红帽只支持带有 Red Hat Identity Management (IdM) 的 FreeRADIUS 作为身份验证的后端源。

在本文档中，RHEL 主机充当一个网桥，以使用现有的网络连接不同的客户端。但是，RHEL 主机只授权认证的客户端可以访问网络。



227\_RHEL\_0422

### 34.1. 先决条件

- 

全新安装 freeradius 和 freeradius-ldap 软件包。

如果软件包已安装，请删除 /etc/raddb/ 目录，卸载，然后再次安装软件包。不要使用 yum reinstall 命令重新安装软件包，因为 /etc/raddb/ 目录中的权限和符号链接会不同。

- 

您在其上配置 FreeRADIUS 的主机是一个 IdM 域 中的客户端。

### 34.2. 在验证器中设置网桥

网桥是一个链路层设备，它根据 MAC 地址表在主机和网络之间转发流量。如果将 RHEL 设置为 802.1X 验证器，请将要在其上执行身份验证的接口和 LAN 接口添加到网桥。

## 前提条件

- 服务器有多个以太网接口。

## 流程

1. 如果网桥接口不存在, 请创建它 :

```
nmcli connection add type bridge con-name br0 ifname br0
```

2. 将太网接口分配给网桥 :

```
nmcli connection add type ethernet slave-type bridge con-name br0-port1 ifname enp1s0 master br0
nmcli connection add type ethernet slave-type bridge con-name br0-port2 ifname enp7s0 master br0
nmcli connection add type ethernet slave-type bridge con-name br0-port3 ifname enp8s0 master br0
nmcli connection add type ethernet slave-type bridge con-name br0-port4 ifname enp9s0 master br0
```

3. 启用网桥以转发 LAN(EAPOL)数据包上的可扩展验证协议 :

```
nmcli connection modify br0 group-forward-mask 8
```

4. 在网桥设备上禁用生成树协议(STP) :

```
*nmcli connection modify br0 stp off"
```

5. 配置连接以自动激活端口 :

```
nmcli connection modify br0 connection.autoconnect-slaves 1
```

6. 激活连接 :

```
nmcli connection up br0
```

## 验证

1.

显示作为特定网桥端口的以太网设备的链接状态：

```
ip link show master br0
3: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
master br0 state UP mode DEFAULT group default qlen 1000
 link/ether 52:54:00:62:61:0e brd ff:ff:ff:ff:ff:ff
...
```

2.

验证 **br0** 设备上是否启用了 **EAPOL** 数据包的转发：

```
cat /sys/class/net/br0/bridge/group_fwd_mask
0x8
```

如果命令返回 **0x8**，则启用了转发。

## 其他资源



[您系统上的 nm-settings \(5\) 手册页](#)

### 34.3. 配置 FREEERADIUS，以使用 EAP 安全地验证网络客户端

FreeRADIUS 支持通过不同的扩展验证协议(EAP)的方法。但是，对于受支持的安全场景，请使用 **EAP-TTLS**（隧道传输层安全）。

使用 **EAP-TTLS**，客户端使用安全 **TLS** 连接作为外部身份验证协议来设置隧道。然后，内部身份验证使用 **LDAP** 向身份管理进行身份验证。要使用 **EAP-TTLS**，您需要一个 **TLS** 服务器证书。



#### 注意

默认的 FreeRADIUS 配置文件充当文档，并描述了所有参数和指令。如果要禁用某些特性，请注释掉它们，而不是删除配置文件中的相应部分。这可让您保留配置文件和包含的文档的结构。

## 先决条件

- 您已安装了 freeradius 和 freeradius-ldap 软件包。
- /etc/raddb/ 目录中的配置文件保持不变，因为是如 freeradius 软件包提供的。
- 主机已在 Red Hat Enterprise Linux Identity Management (IdM) 域中注册。

## 流程

- 创建一个私钥并从 IdM 请求一个证书：

```
ipa-getcert request -w -k /etc/pki/tls/private/radius.key -f /etc/pki/tls/certs/radius.pem
-o "root:radiusd" -m 640 -O "root:radiusd" -M 640 -T calPAserviceCert -C 'systemctl
restart radiusd.service' -N freeradius.idm.example.com -D freeradius.idm.example.com
-K radius/freeradius.idm.example.com
```

certmonger 服务将私钥存储在 /etc/pki/tls/private/radius.key 文件中，将证书存储在 /etc/pki/tls/certs/radius.pem 文件中，并设置安全权限。另外，certmonger 将监控证书，在证书过期前续订证书，并在证书续订后重启 radiusd 服务。

- 验证 CA 是否已成功发布证书：

```
ipa-getcert list -f /etc/pki/tls/certs/radius.pem
...
Number of certificates and requests being tracked: 1.
Request ID '20240918142211':
 status: MONITORING
 stuck: no
 key pair storage: type=FILE,location='/etc/pki/tls/private/radius.key'
 certificate: type=FILE,location='/etc/pki/tls/certs/radius.crt'
 ...
 ...
```

- 使用 Diffie-Hellman (DH) 参数创建 /etc/raddb/certs/dh 文件。例如，要创建带有 2048 位素数的 DH 文件，请输入：

```
openssl dhparam -out /etc/raddb/certs/dh 2048
```

为了安全起见，请不要使用小于 2048 位素数的 DH 文件。根据位数，文件的创建可能需要几分钟。

4.

编辑 `/etc/raddb/mods-available/eap` 文件：

a.

在 `tls-config tls-common` 指令中配置与 TLS 相关的设置：

```
eap {
 ...
 tls-config tls-common {
 ...
 private_key_file = /etc/pki/tls/private/radius.key
 certificate_file = /etc/pki/tls/certs/radius.pem
 ca_file = /etc/ipa/ca.crt
 ...
 }
}
```

b.

将 `eap` 指令中的 `default_eap_type` 参数设为 `ttls`：

```
eap {
 ...
 default_eap_type = ttls
 ...
}
```

c.

注释掉 `md5` 指令，以禁用不安全的 EAP-MD5 身份验证方法：

```
eap {
 ...
 # md5 {
 # }
 ...
}
```

请注意，在默认的配置文件中，其他不安全的 EAP 身份验证方法默认被注释掉了。

5.

编辑 `/etc/raddb/sites-available/default` 文件，然后注释掉 `eap` 以外的所有身份验证方法：

```
authenticate {
 ...
 # Auth-Type PAP {
 # pap
```

```

}

Auth-Type CHAP {
chap
}

Auth-Type MS-CHAP {
mschap
}

mschap

digest
...
}

```

这将只对外部身份验证启用了 EAP，并禁用了纯文本身份验证方法。

6.

编辑 /etc/raddb/sites-available/inner-tunnel 文件，并进行以下更改：

a.

注释掉 -ldap 条目，并将 ldap 模块配置添加到 authorize 指令中：

```

authorize {
...
#-ldap
ldap
if ((ok || updated) && User-Password) {
 update {
 control:Auth-Type := ldap
 }
}

...
}

```

b.

在 authenticate 指令中取消 LDAP 身份验证类型的注释：

```

authenticate {
 Auth-Type LDAP {
 ldap
 }
}

```

7.

启用 ldap 模块：

```
ln -s /etc/raddb/mods-available/ldap /etc/raddb/mods-enabled/ldap
```

8.

编辑 `/etc/raddb/mods-available/ldap` 文件，并进行以下更改：

a.

在 `ldap` 指令中，设置 IdM LDAP 服务器 URL 和基本可分辨名称(DN)：

```
ldap {
 ...
 server = 'ldaps://idm_server.idm.example.com'
 base_dn = 'cn=users,cn=accounts,dc=idm,dc=example,dc=com'
 ...
}
```

在服务器 URL 中指定 `ldaps` 协议，以便在 FreeRADIUS 主机和 IdM 服务器之间使用 TLS 加密的连接。

b.

在 `ldap` 指令中，启用 IdM LDAP 服务器的 TLS 证书验证：

```
tls {
 ...
 require_cert = 'demand'
 ...
}
```

9.

编辑 `/etc/raddb/clients.conf` 文件：

a.

在 `localhost` 和 `localhost_ipv6` 客户端指令中设置安全密码：

```
client localhost {
 ipaddr = 127.0.0.1
 ...
 secret = localhost_client_password
 ...
}

client localhost_ipv6 {
 ipv6addr = ::1
 secret = localhost_client_password
}
```

b.

为网络验证器添加客户端指令：

```
client hostapd.example.org {
 ipaddr = 192.0.2.2/32
 secret = hostapd_client_password
}
```

c.

可选：如果其他主机也能够访问 FreeRADIUS 服务，也为它们添加客户端指令，例如：

```
client <hostname_or_description>{
 ipaddr = <IP_address_or_range>
 secret = <client_password>
}
```

**ipaddr** 参数接受 IPv4 和 IPv6 地址，您可以使用可选的无类别域间路由(CIDR)表示法来指定范围。但是，在这个参数中您只能设置一个值。例如，要授予对 IPv4 和 IPv6 地址的访问，您必须添加两个客户端指令。

为客户端指令使用一个描述性名称，如主机名或一个描述 IP 范围在哪里使用的词语。

10.

验证配置文件：

```
radiusd -XC
...
Configuration appears to be OK
```

11.

在 firewalld 服务中打开 RADIUS 端口：

```
firewall-cmd --permanent --add-service=radius
firewall-cmd --reload
```

12.

启用并启动 radiusd 服务：

```
systemctl enable --now radiusd
```

验证

- 针对 FreeRADIUS 服务器或验证器测试 EAP-TTLS 身份验证

## 故障排除

- 如果 radiusd 服务无法启动, 请验证您是否可以解析 IdM 服务器主机名 :

```
host -v idm_server.idm.example.com
```

- 对于其他问题, 请在 debug 模式下运行 radiusd :

a.

停止 radiusd 服务 :

```
systemctl stop radiusd
```

b.

以 debug 模式启动该服务 :

```
radiusd -X
...
Ready to process requests
```

c.

在 FreeRADIUS 主机上执行验证测试, 如 验证 部分中所述。

## 后续步骤

- 禁用不再需要的身份验证方法, 以及您不使用的其他功能。

### 34.4. 在有线网络中将 HOSTAPD 配置为验证器

主机访问点守护进程 (hostapd) 服务可在有线网络中充当验证器, 来提供 802.1X 身份验证。为此, hostapd 服务需要一个用来验证客户端的 RADIUS 服务器。

hostapd 服务提供集成的 RADIUS 服务器。但是, 使用集成的 RADIUS 服务器只用于测试目的。对于生产环境, 请使用 FreeRADIUS 服务器, 它支持其他特性, 如不同的身份验证方法和访问控制。



## 重要

**hostapd** 服务不与流量平面交互。该服务仅充当身份验证器。例如，使用脚本或服务，该脚本或服务使用 **hostapd** 控制接口、根据身份验证事件的结果来允许或拒绝流量。

## 先决条件

- **hostapd** 软件包已安装。
- FreeRADIUS 服务器已配置，它已准备好对客户端进行身份验证。

## 流程

1. 使用以下内容创建 /etc/hostapd/hostapd.conf 文件：

```
General settings of hostapd
=====

Control interface settings
ctrl_interface=/var/run/hostapd
ctrl_interface_group=wheel

Enable logging for all modules
logger_syslog=-1
logger_stdout=-1

Log level
logger_syslog_level=2
logger_stdout_level=2

Wired 802.1X authentication
=====

Driver interface type
driver=wired

Enable IEEE 802.1X authorization
ieee8021x=1

Use port access entry (PAE) group address
(01:80:c2:00:00:03) when sending EAPOL frames
use_pae_group_addr=1

Network interface for authentication requests
interface=br0
```

```
RADIUS client configuration
=====

Local IP address used as NAS-IP-Address
own_ip_addr=192.0.2.2

Unique NAS-Identifier within scope of RADIUS server
nas_identifier=hostapd.example.org

RADIUS authentication server
auth_server_addr=192.0.2.1
auth_server_port=1812
auth_server_shared_secret=hostapd_client_password

RADIUS accounting server
acct_server_addr=192.0.2.1
acct_server_port=1813
acct_server_shared_secret=hostapd_client_password
```

有关此配置中使用的参数的详情，请查看 `/usr/share/doc/hostapd/hostapd/hostapd.conf` 示例配置文件中的描述。

2.

启用并启动 `hostapd` 服务：

```
systemctl enable --now hostapd
```

验证

- 

[针对 FreeRADIUS 服务器或验证器测试 EAP-TTLS 身份验证](#)

故障排除

- 

[如果 `hostapd` 服务无法启动，请验证您在 `/etc/hostapd/hostapd.conf` 文件中使用的网桥接口是否在系统上存在：](#)

```
ip link show br0
```

- 

[对于其他问题，请在 `debug` 模式下运行 `hostapd`：](#)

a.

[停止 `hostapd` 服务：](#)

```
systemctl stop hostapd
```

b.

以 **debug** 模式启动该服务：

```
hostapd -d /etc/hostapd/hostapd.conf
```

c.

在 FreeRADIUS 主机上执行验证测试，如 验证 部分中所述。

## 其他资源

- 您系统上的 **hostapd.conf (5)** 手册页
- **/usr/share/doc/hostapd/hostapd.conf** 文件

### 34.5. 针对 FREERADIUS 服务器或验证器测试 EAP-TTLS 身份验证

要测试在隧道传输层安全(EAP-TTLS)上使用可扩展身份验证协议(EAP-TTLS)的身份验证是否按预期工作，请运行此流程：

- 设置 FreeRADIUS 服务器后
- 将 hostapd 服务设为 802.1X 网络身份验证验证器后。

此流程中使用的测试工具的输出提供有关 EAP 通信的其他信息，并帮助您调试问题。

## 先决条件

- 当您要验证：
  - FreeRADIUS 服务器：

- hostapd 软件包提供的 eapol\_test 工具已安装。
- 您在其上运行此流程的客户端已在 FreeRADIUS 服务器的客户端数据库中被授权。
  - 由同名软件包提供的验证器 wpa\_supplicant 工具已安装。
- 您在 /etc/ipa/ca.cert 文件中存储了证书颁发机构(CA)证书。

## 流程

1. 可选：在 Identity Management (IdM) 中创建用户：

```
ipa user-add --first "Test" --last "User" idm_user --password
```

2. 使用以下内容创建 /etc/wpa\_supplicant/wpa\_supplicant-TTLS.conf 文件：

```
ap_scan=0

network={
 eap=TTLS
 eapol_flags=0
 key_mgmt=IEEE8021X

 # Anonymous identity (sent in unencrypted phase 1)
 # Can be any string
 anonymous_identity="anonymous"

 # Inner authentication (sent in TLS-encrypted phase 2)
 phase2="auth=PAP"
 identity="idm_user"
 password="idm_user_password"

 # CA certificate to validate the RADIUS server's identity
 ca_cert="/etc/ipa/ca.crt"
}
```

3. 要向以下进行身份验证：

- FreeRADIUS 服务器，请输入：

```
eapol_test -c /etc/wpa_supplicant/wpa_supplicant-TTLS.conf -a 192.0.2.1 -s <client_password>
...
EAP: Status notification: remote certificate verification (param=success)
...
CTRL-EVENT-EAP-SUCCESS EAP authentication completed successfully
...
SUCCESS
```

-a 选项定义了 FreeRADIUS 服务器的 IP 地址，而 -s 选项指定您要在其上运行 FreeRADIUS 服务器的客户端配置中命令的主机的密码。

- 验证器，请输入：

```
wpa_supplicant -c /etc/wpa_supplicant/wpa_supplicant-TTLS.conf -D wired -i enp0s31f6
...
enp0s31f6: CTRL-EVENT-EAP-SUCCESS EAP authentication completed successfully
...
```

-i 选项指定 wpa\_supplicant 通过 LAN(EAPOL)数据包发送扩展验证协议的网络接口名称。

若要了解更多调试信息，请将 -d 选项传给命令。

## 其他资源

- /usr/share/doc/wpa\_supplicant/wpa\_supplicant.conf 文件

### 34.6. 根据 HOSTAPD 身份验证事件阻止和允许流量

hostapd 服务不与流量平面交互。该服务仅充当身份验证器。但是，您可以编写一个脚本，根据身份验证事件的结果来允许或拒绝流量。

**重要**

此流程不受支持，没有企业级的解决方案。它只演示如何通过评估由 hostapd\_cli 检索的事件来阻止或允许流量。

当 802-1x-tr-mgmt systemd 服务启动时，RHEL 会阻止 hostapd 监听端口上的所有流量，但 LAN(EAPOL)数据包上可扩展验证协议除外，并使用 hostapd\_cli 工具连接到 hostapd 控制接口。/usr/local/bin/802-1x-tr-mgmt 脚本随后评估事件。根据 hostapd\_cli 收到的不同事件，该脚本允许或阻止 MAC 地址的流量。请注意，当 802-1x-tr-mgmt 服务停止时，所有流量会自动允许。

在 hostapd 服务器上执行此流程。

**先决条件**

- hostapd 服务已配置，服务已准备好对客户端进行身份验证。

**流程**

1. 使用以下内容创建 /usr/local/bin/802-1x-tr-mgmt 文件：

```
#!/bin/sh

TABLE="tr-mgmt-${1}"
read -r -d '' TABLE_DEF << EOF
table bridge ${TABLE} {
 set allowed_macs {
 type ether_addr
 }

 chain accesscontrol {
 ether saddr @allowed_macs accept
 ether daddr @allowed_macs accept
 drop
 }

 chain forward {
 type filter hook forward priority 0; policy accept;
 meta ibrname "$1" jump accesscontrol
 }
}
EOF

case ${2:-NOTANEVENT} in
 block_all)
```

```

nft destroy table bridge "$TABLE"
printf "$TABLE_DEF" | nft -f -
echo "$1: All the bridge traffic blocked. Traffic for a client with a given MAC will
be allowed after 802.1x authentication"
;

AP-STA-CONNECTED | CTRL-EVENT-EAP-SUCCESS | CTRL-EVENT-EAP-
SUCCESS2)
nft add element bridge tr-mgmt-br0 allowed_macs { $3 }
echo "$1: Allowed traffic from $3"
;

AP-STA-DISCONNECTED | CTRL-EVENT-EAP-FAILURE)
nft delete element bridge tr-mgmt-br0 allowed_macs { $3 }
echo "$1: Denied traffic from $3"
;

allow_all)
nft destroy table bridge "$TABLE"
echo "$1: Allowed all bridge traffic again"
;

NOTANEVENT)
echo "$0 was called incorrectly, usage: $0 interface event [mac_address]"
;
esac

```

2.

使用以下内容创建 /etc/systemd/system/802-1x-tr-mgmt@.service systemd 服务文件：

```

[Unit]
Description=Example 802.1x traffic management for hostapd
After=hostapd.service
After=sys-devices-virtual-net-%i.device

[Service]
Type=simple
ExecStartPre=bash -c '/usr/sbin/hostapd_cli ping | grep PONG'
ExecStartPre=/usr/local/bin/802-1x-tr-mgmt %i block_all
ExecStart=/usr/sbin/hostapd_cli -i %i -a /usr/local/bin/802-1x-tr-mgmt
ExecStopPost=/usr/local/bin/802-1x-tr-mgmt %i allow_all

[Install]
WantedBy=multi-user.target

```

3.

重新载入 systemd：

```
systemctl daemon-reload
```

4.

启用并启动接口名称 hostapd 正在侦听的 802-1x-tr-mgmt 服务：

```
systemctl enable --now 802-1x-tr-mgmt@br0.service
```

## 验证

- 通过客户端向网络进行身份验证。请参阅 [针对 FreeRADIUS 服务器或验证器测试 EAP-TTLS 身份验证](#)。

## 其他资源

- 您系统上的 [systemd.service \(5\)](#) 手册页

## 第 35 章 多路径 TCP 入门

传输控制协议(TCP)确保通过互联网的可靠的数据传递，并自动调整其带宽以响应网络负载。多路径 TCP (MPTCP) 是原始 TCP 协议 (single-path) 的扩展。MPTCP 支持传输连接同时在多个路径中运行，并为用户端点设备带来网络连接冗余。

### 35.1. 了解 MPTCP

多路径 TCP (MPTCP) 协议允许在连接端点间同时使用多个路径。协议设计提高了连接稳定性，与单一路径 TCP 相比也带来了其他好处。



#### 注意

在 MPTCP 术语中，链接被视为路径。

以下是使用 MPTCP 的一些优点：

- 它允许一个连接同时使用多个网络接口。
- 如果连接绑定到链路速度，则使用多个链接可能会增加连接的吞吐量。请注意，如果连接绑定到 CPU，则使用多个链路会导致连接性能下降。
- 它提高对链接故障的恢复能力。

有关 MPTCP 的详情，请查看 [其他资源](#)。

#### 其他资源

- [了解多路径 TCP：端点和高可用性以及未来的网络](#)
- [RFC8684：通过多个地址进行多路径操作的 TCP 扩展](#)

- 

## Red Hat Enterprise Linux 8.3 上的多路径 TCP：从 0 到 1 个子流

### 35.2. 准备 RHEL 启用 MPTCP 支持

默认情况下，RHEL 中禁用 MPTCP 支持。启用 MPTCP，以便支持此特性的应用程序可以使用它。此外，您必须配置用户空间应用程序，以便在那些应用程序默认具有 TCP 套接字时强制使用 MPTCP 套接字。

您可以使用 `sysctl` 工具启用 MPTCP 支持，并使用 SystemTap 脚本为系统范围的应用程序启用 MPTCP 准备 RHEL。

#### 前提条件

安装以下软件包：

- 

`systemtap`

- 

`iperf3`

#### 流程

1.

在内核中启用 MPTCP 套接字：

```
echo "net.mptcp.enabled=1" > /etc/sysctl.d/90-enable-MPTCP.conf
sysctl -p /etc/sysctl.d/90-enable-MPTCP.conf
```

2.

验证内核中是否启用了 MPTCP：

```
sysctl -a | grep mptcp.enabled
net.mptcp.enabled = 1
```

3.

使用以下内容创建 `mptcp-app.stap` 文件：

```
#!/usr/bin/env stap
```

```
%{
#include <linux/in.h>
#include <linux/ip.h>
%}

/* RSI contains 'type' and RDX contains 'protocol'.
 */

function mptcpify () %{
 if (CONTEXT->kregs->si == SOCK_STREAM &&
 (CONTEXT->kregs->dx == IPPROTO_TCP ||
 CONTEXT->kregs->dx == 0)) {
 CONTEXT->kregs->dx = IPPROTO_MPTCP;
 STAP_RETVALUE = 1;
 } else {
 STAP_RETVALUE = 0;
 }
%}

probe kernel.function("__sys_socket") {
 if (mptcpify() == 1) {
 printf("command %16s mptcpified\n", execname());
 }
}
```

4.

强制用户空间应用程序创建 MPTCP 套接字而不是 TCP 套接字：

```
stap -vg mptcp-app.stap
```

注意：此操作会影响命令之后启动的所有 TCP 套接字。在使用 Ctrl+C 中断上述命令后，应用将继续使用 TCP 套接字。

5.

另外，要允许 MPTCP 只用于特定的应用程序，您可以使用以下内容修改 mptcp-app.stap 文件：

```
#!/usr/bin/env stap

%{
#include <linux/in.h>
#include <linux/ip.h>
%}

/* according to [1], RSI contains 'type' and RDX
 * contains 'protocol'.
 * [1] https://github.com/torvalds/linux/blob/master/arch/x86/entry/entry_64.S#L79
 */

function mptcpify () %{
 if (CONTEXT->kregs->si == SOCK_STREAM &&
```

```

(CONTEXT->kregs->dx == IPPROTO_TCP ||
 CONTEXT->kregs->dx == 0)) {
CONTEXT->kregs->dx = IPPROTO_MPTCP;
STAP_RETVALUE = 1;
} else {
STAP_RETVALUE = 0;
}
%}

probe kernel.function("__sys_socket") {
cur_proc = execname()
if ((cur_proc == @1) && (mptcpify() == 1)) {
printf("command %16s mptcpified\n", cur_proc);
}
}

```

6.

在可选择的情况下，假设您想要强制 iperf3 工具使用 MPTCP 而不是 TCP。要做到这一点，请输入以下命令：

```
stap -vg mptcp-app.stap iperf3
```

7.

在 mptcp-app.stap 脚本安装内核探测后，会在内核 dmesg 输出中出现以下警告

```

dmesg
...
[1752.694072] Kprobes globally unoptimized
[1752.730147] stap_1ade3b3356f3e68765322e26dec00c3d_1476: module_layout:
kernel tainted.
[1752.732162] Disabling lock debugging due to kernel taint
[1752.733468] stap_1ade3b3356f3e68765322e26dec00c3d_1476: loading out-of-tree
module taints kernel.
[1752.737219] stap_1ade3b3356f3e68765322e26dec00c3d_1476: module verification
failed: signature and/or required key missing - tainting kernel
[1752.737219] stap_1ade3b3356f3e68765322e26dec00c3d_1476 (mptcp-app.stap):
systemtap: 4.5/0.185, base: ffffffc0550000, memory:
224data/32text/57ctx/65638net/367alloc kb, probes: 1

```

8.

启动 iperf3 服务器：

```

iperf3 -s
Server listening on 5201

```

9.

将客户端连接到服务器：

```
iperf3 -c 127.0.0.1 -t 3
```

10.

建立连接后，验证 **ss** 输出以查看特定于子流的状态：

```
ss -nti '(dport :5201)'

State Recv-Q Send-Q Local Address:Port Peer Address:Port Process
ESTAB 0 0 127.0.0.1:41842 127.0.0.1:5201
cubic wscale:7,7 rto:205 rtt:4.455/8.878 ato:40 mss:21888 pmtu:65535 rcvmss:536
advmss:65483 cwnd:10 bytes_sent:141 bytes_acked:142 bytes_received:4 segs_out:8
segs_in:7 data_segs_out:3 data_segs_in:3 send 393050505bps lastsnd:2813
lastrcv:2772 lastack:2772 pacing_rate 785946640bps delivery_rate 10944000000bps
delivered:4 busy:41ms rcv_space:43690 rcv_ssthresh:43690 minrtt:0.008 tcp-ulp-
mptcp flags:Mmec token:0000(id:0)/2ff053ec(id:0) seq:3e2cbea12d7673d4 sfseq:3
ssnoff:ad3d00f4 maplen:2
```

11.

验证 MPTCP 计数器：

```
nstat MPTcp*
#kernel
MPTcpExtMPCapableSYNRX 2 0.0
MPTcpExtMPCapableSYNTX 2 0.0
MPTcpExtMPCapableSYNACKRX 2 0.0
MPTcpExtMPCapableACKRX 2 0.0
```

## 其他资源

- [如何为 RHEL 系统下载或安装 debuginfo 软件包？（红帽知识库）](#)
- 您系统上的 **tcp(7)** 和 **mptcpize (8)** 手册页

### 35.3. 使用 IPROUTE2 为 MPTCP 应用程序临时配置和启用多个路径

每个 MPTCP 连接都使用一个类似于纯 TCP 的单个子流。要获得 MPTCP 好处，请为每个 MPTCP 连接的最大子流数指定高些的限制。然后配置额外的端点以创建这些子流。



#### 重要

在重启机器后，这个过程中的配置不会保留。

请注意，MPTCP 尚不支持为同一套接字混合 IPv6 和 IPv4 端点。使用属于同一地址系列的端点。

## 先决条件

- iperf3 软件包已安装
- 服务器网络接口设置：
  - enp4s0: 192.0.2.1/24
  - enp1s0: 198.51.100.1/24
- 客户端网络接口设置：
  - enp4s0f0: 192.0.2.2/24
  - enp4s0f1: 198.51.100.2/24

## 步骤

1. 将客户端配置为接受最多 1 个额外的远程地址，如服务器提供的地址：

```
ip mptcp limits set add_addr_accepted 1
```

2. 在服务器上将 IP 地址 198.51.100.1 添加为新的 MPTCP 端点：

```
ip mptcp endpoint add 198.51.100.1 dev enp1s0 signal
```

**signal** 选项可确保在三向手后发送 ADD\_ADDR 数据包。

3.

启动 iperf3 服务器：

```
iperf3 -s
Server listening on 5201
```

4.

将客户端连接到服务器：

```
iperf3 -c 192.0.2.1 -t 3
```

## 验证

1.

验证连接是否已建立：

```
ss -nti '(sport :5201)'
```

2.

验证连接和 IP 地址限制：

```
ip mptcp limit show
```

3.

验证新添加的端点：

```
ip mptcp endpoint show
```

4.

在服务器上使用 nstat MPTcp\* 命令验证 MPTCP 计数器：

```
nstat MPTcp*

#kernel
MPTcpExtMPCapableSYNRX 2 0.0
MPTcpExtMPCapableACKRX 2 0.0
MPTcpExtMPJoinSynRx 2 0.0
MPTcpExtMPJoinAckRx 2 0.0
MPTcpExtEchoAdd 2 0.0
```

## 其他资源

- 

您系统上的 mptcpize (8) 和 ip-mptcp (8) 手册页

### 35.4. 为 MPTCP 应用程序永久配置多路径

您可以使用 `nmcli` 命令配置 MultiPath TCP (MPTCP)，以在源和目标系统之间永久建立多个子流。子流可以使用不同的资源，不同的路由到目的地，甚至不同网络。例如以太网、单元格、wifi 等。因此，您可以实现组合连接，这提高了网络弹性和吞吐量。

服务器在我们的示例中使用以下网络接口：

- `enp4s0: 192.0.2.1/24`
- `enp1s0: 198.51.100.1/24`
- `enp7s0: 192.0.2.3/24`

客户端在我们的示例中使用以下网络接口：

- `enp4s0f0: 192.0.2.2/24`
- `enp4s0f1: 198.51.100.2/24`
- `enp6s0: 192.0.2.5/24`

#### 先决条件

- 您在相关接口上配置了默认网关。

#### 流程

1. 在内核中启用 MPTCP 套接字：

```
echo "net.mptcp.enabled=1" > /etc/sysctl.d/90-enable-MPTCP.conf
sysctl -p /etc/sysctl.d/90-enable-MPTCP.conf
```

2.

可选：用于子流限制的 RHEL 内核默认是 2。如果您需要更多：

a.

使用以下内容创建 `/etc/systemd/system/set_mptcp_limit.service` 文件：

```
[Unit]
Description=Set MPTCP subflow limit to 3
After=network.target

[Service]
ExecStart=ip mptcp limits set subflows 3
Type=oneshot

[Install]
WantedBy=multi-user.target
```

在每个引导过程中，当网络 (`network.target`) 可以正常工作后，`oneshot` 单元会执行 `ip mptcp limits set subflows 3` 命令。

b.

启用 `set_mptcp_limit` 服务：

```
systemctl enable --now set_mptcp_limit
```

3.

在要用于连接聚合的所有连接配置集上启用 MPTCP：

```
nmcli connection modify <profile_name> connection.mptcp-flags
signal,subflow,also-without-default-route
```

`connection.mptcp-flags` 参数配置 MPTCP 端点和 IP 地址标志。如果在 NetworkManager 连接配置集中启用 MPTCP，则设置会将相关网络接口的 IP 地址配置为 MPTCP 端点。

默认情况下，如果没有默认网关，NetworkManager 不会将 MPTCP 标记添加到 IP 地址。如果要绕过该检查，您需要使用 `also-without-default-route` 标志。

## 验证

1.

验证您启用了 MPTCP 内核参数：

```
sysctl net.mptcp.enabled
net.mptcp.enabled = 1
```

2.

验证您正确设置了子流限制，如果默认值是不够的：

```
ip mptcp limit show
add_addr_accepted 2 subflows 3
```

3.

验证您正确配置了每个地址 MPTCP 设置：

```
ip mptcp endpoint show
192.0.2.1 id 1 subflow dev enp4s0
198.51.100.1 id 2 subflow dev enp1s0
192.0.2.3 id 3 subflow dev enp7s0
192.0.2.4 id 4 subflow dev enp3s0
...
```

## 其他资源

- [nm-settings-nmcli\(5\)](#)
- [ip-mptcp\(8\)](#)
- [第 35.1 节 “了解 MPTCP”](#)
- [了解多路径 TCP：端点和高可用性以及未来的网络](#)
- [RFC8684：通过多个地址进行多路径操作的 TCP 扩展](#)

- 使用多路径 TCP 更好地停机并增加带宽

### 35.5. 监控 MPTCP 子流

多路径 TCP(MPTCP)套接字的生命周期可能比较复杂：创建主 MPTCP 套接字，验证 MPTCP 路径，创建一个或多个子流并最终删除。最后，终止 MPTCP 套接字。

MPTCP 协议允许使用 iproute 软件包提供的 ip 工具监控与套接字和子流创建和删除相关的特定于 MPTCP 的事件。这个工具使用 netlink 接口来监控 MPTCP 事件。

此流程演示了如何监控 MPTCP 事件。因此，它会模拟 MPTCP 服务器应用程序，以及客户端连接到此服务。本例中涉及的客户端使用以下接口和 IP 地址：

- 服务器：192.0.2.1
- 客户端（以太网连接）：192.0.2.2
- 客户端（WiFi 连接）：192.0.2.3

为了简化这一示例，所有接口都在同一子网内。这不是必须的。但是，重要的是路由已正确配置，并且客户端能够通过两个接口访问服务器。

#### 前提条件

- 有两个网络接口的 RHEL 客户端，如带有以太网和 WiFi 的笔记本电脑
- 客户端可以通过两个接口连接到服务器
- RHEL 服务器

- 客户端和服务器都运行 RHEL 8.6 或更高版本

## 流程

1. 将客户端和服务器上的每个连接额外子流的限制设为 1：

```
ip mptcp limits set add_addr_accepted 0 subflows 1
```

2. 在服务器上，要模拟 MPTCP 服务器应用程序，请使用强制的 MPTCP 套接字而不是 TCP 套接字，以侦听模式启动 netcat (nc)：

```
nc -l -k -p 12345
```

-k 选项可使 nc 在第一次接受连接后不关闭监听程序。这要求演示子流的监控。

3. 在客户端中：

- a. 识别具有最低指标的接口：

```
ip -4 route
192.0.2.0/24 dev enp1s0 proto kernel scope link src 192.0.2.2 metric 100
192.0.2.0/24 dev wlp1s0 proto kernel scope link src 192.0.2.3 metric 600
```

enp1s0 接口具有比 wlp1s0 更低的指标。因此，RHEL 默认使用 enp1s0。

- b. 在第一个终端上，启动监控：

```
ip mptcp monitor
```

- c. 在第二个终端上，启动到服务器的 MPTCP 连接：

```
nc 192.0.2.1 12345
```

RHEL 使用 `enp1s0` 接口及其关联的 IP 地址作为此连接的源。

在监控终端上，`ip mptcp monitor` 命令现在记录：

```
[CREATED] token=63c070d2 remid=0 locid=0 saddr4=192.0.2.2
daddr4=192.0.2.1 sport=36444 dport=12345
```

这个令牌将 MPTCP 套接字标识为唯一的 ID，之后它会在同一套接字上关联 MPTCP 事件。

d.

在运行 `nc` 连接到服务器的终端上，按 `Enter` 键。第一个数据包完全建立连接。请注意，只要没有发送数据，连接就不会建立。

在监控终端上，`ip mptcp monitor` 现在记录日志：

```
[ESTABLISHED] token=63c070d2 remid=0 locid=0 saddr4=192.0.2.2
daddr4=192.0.2.1 sport=36444 dport=12345
```

e.

可选：显示到服务器上端口 12345 的连接：

```
ss -taunp | grep ":12345"
tcp ESTAB 0 0 192.0.2.2:36444 192.0.2.1:12345
```

此时，只建立一个到服务器的连接。

f.

在第三个终端上，创建另一个端点：

```
ip mptcp endpoint add dev wlp1s0 192.0.2.3 subflow
```

此命令设置客户端 WiFi 接口的名称和 IP 地址。

在监控终端上，`ip mptcp monitor` 现在记录日志：

```
[SF_ESTABLISHED] token=63c070d2 remid=0 locid=2 saddr4=192.0.2.3
```

```
[daddr4=192.0.2.1 sport=53345 dport=12345 backup=0 ifindex=3]
```

**locid** 字段显示新子流的本地地址 ID，即使连接使用了网络地址转换(NAT)，也标识此子流。**saddr4** 字段与 **ip mptcp endpoint add** 命令的端点的 IP 地址匹配。

g.

可选：显示到服务器上端口 12345 的连接：

```
ss -taunp | grep ":12345"
tcp ESTAB 0 0 192.0.2.2:36444 192.0.2.1:12345
tcp ESTAB 0 0 192.0.2.3%wlp1s0:53345 192.0.2.1:12345
```

该命令现在显示两个连接：

- 源地址为 192.0.2.2 的、与之前建立的第一个 MPTCP 子流相对应的连接。
- 来自 wlp1s0 接口及源地址 192.0.2.3 的子流的连接。

h.

在第三个终端上，删除端点：

```
ip mptcp endpoint delete id 2
```

使用 **ip mptcp monitor** 输出中的 **locid** 字段的 ID，或者使用 **ip mptcp endpoint show** 命令来检索端点 ID。

在监控终端上，**ip mptcp monitor** 现在记录日志：

```
[SF_CLOSED] token=63c070d2 remid=0 locid=2 saddr4=192.0.2.3
daddr4=192.0.2.1 sport=53345 dport=12345 backup=0 ifindex=3
```

i.

在第一个带有 **nc** 客户端的终端上，按 **Ctrl+C** 来终止会话。

在监控终端上，**ip mptcp monitor** 现在记录日志：

```
[CLOSED] token=63c070d2
```

## 其他资源

- 您系统上的 **ip-mptcp (1)** 手册页
- [NetworkManager 如何管理多个默认网关](#)

### 35.6. 在内核中禁用多路径 TCP

您可以在内核中明确禁用 MPTCP 选项。

#### 流程

- 禁用 mptcp.enabled 选项。

```
echo "net.mptcp.enabled=0" > /etc/sysctl.d/90-enable-MPTCP.conf
sysctl -p /etc/sysctl.d/90-enable-MPTCP.conf
```

#### 验证

- 验证是否在内核中禁用了 mptcp.enabled 。

```
sysctl -a | grep mptcp.enabled
net.mptcp.enabled = 0
```

## 第 36 章 RHEL 中支持旧的网络脚本

默认情况下，RHEL 使用 NetworkManager 配置和管理网络连接，`/usr/sbin/ifup` 和 `/usr/sbin/ifdown` 脚本使用 NetworkManager 来处理 `/etc/sysconfig/network-scripts/` 目录中的 `ifcfg` 文件。



### 重要

旧脚本在 RHEL 8 中已弃用，并将在以后的 RHEL 主要版本中被删除。如果您仍然使用旧的网络脚本，例如，因为您从较早的版本升级到 RHEL 8，红帽建议将您的配置迁移至 NetworkManager。

### 36.1. 安装旧的网络脚本

如果您需要使用弃用的网络脚本，而不是使用 NetworkManager 来处理网络配置，您可以安装它们。在这种情况下，`/usr/sbin/ifup` 和 `/usr/sbin/ifdown` 脚本链接到管理网络配置的已弃用的 shell 脚本。

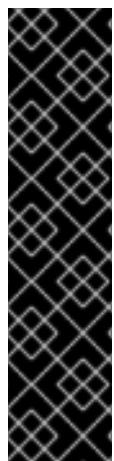
#### 流程

- 安装 `network-scripts` 软件包：

```
yum install network-scripts
```

## 第 37 章 使用 IFCFG 文件配置 IP 网络

接口配置(ifcfg)文件控制各个网络设备的软件接口。当系统引导时，它使用这些文件来决定启动哪些界面以及如何进行配置。这些文件被命名为 `ifcfg-name_pass`，其中后缀 `name` 指的是配置文件控制的设备的名称。按照惯例，`ifcfg` 文件的后缀与配置文件中 `DEVICE` 指令给出的字符串相同。



### 重要

`NetworkManager` 支持以密钥文件格式存储的配置文件。但是，当使用 `NetworkManager API` 创建或更新配置文件时，`NetworkManager` 默认使用 `ifcfg` 格式。

在未来的主 RHEL 版本中，密钥文件格式将会成为默认格式。如果要手动创建和管理配置文件，请考虑使用密钥文件格式。详情请查看 [keyfile 格式的 NetworkManager 连接配置文件](#)。

### 37.1. 使用 IFCFG 文件配置带有静态网络设置的接口

如果不使用 `NetworkManager` 工具和应用程序，您可以通过创建 `ifcfg` 文件来手动配置网络接口。

#### 流程

- 要使用 `ifcfg` 文件，为名为 `enp1s0` 的接口配置具有静态网络设置的接口，请在 `/etc/sysconfig/network-scripts/` 目录中创建一个名为 `ifcfg-enp1s0` 的文件，其包含以下内容：
  - 对于 IPv4 配置：

```
DEVICE=enp1s0
BOOTPROTO=none
ONBOOT=yes
PREFIX=24
IPADDR=192.0.2.1
GATEWAY=192.0.2.254
```

- 对于 IPv6 配置：

```
DEVICE=enp1s0
BOOTPROTO=none
ONBOOT=yes
```

```
IPV6INIT=yes
IPV6ADDR=2001:db8:1::2/64
```

## 其他资源

- 您系统上的 **nm-settings-ifcfg-rh (5)** 手册页

### 37.2. 使用 IFCFG 文件配置带有动态网络设置的接口

如果不使用 NetworkManager 工具和应用程序，您可以通过创建 **ifcfg** 文件来手动配置网络接口。

## 流程

1. 要使用 **ifcfg** 文件配置具有动态网络配置、名为 **em1** 的接口，请在 **/etc/sysconfig/network-scripts/** 目录中创建一个名为 **ifcfg-em1** 的文件，其包含以下内容：

```
DEVICE=em1
BOOTPROTO=dhcp
ONBOOT=yes
```

2. 要配置发送的接口：
  - DHCP 服务器使用不同的主机名，请在 **ifcfg** 文件中添加以下行：

```
DHCP_HOSTNAME=hostname
```

- DHCP 服务器使用不同的完全限定域名(FQDN)，请在 **ifcfg** 文件中添加以下行：

```
DHCP_FQDN=fully.qualified.domain.name
```



#### 注意

您只能使用这些设置中的一个。如果您同时指定了 **DHCP\_HOSTNAME** 和 **DHCP\_FQDN**，则只使用 **DHCP\_FQDN**。

3.

要将接口配置为使用特定的 DNS 服务器, 请在 `ifcfg` 文件中添加以下行 :

```
PEERDNS=no
DNS1=ip-address
DNS2=ip-address
```

其中 *ip-address* 是 DNS 服务器的地址。这会导致网络服务使用指定的 DNS 服务器更新 `/etc/resolv.conf`。只需要一个 DNS 服务器地址, 另一个是可选的。

### 37.3. 使用 IFCFG 文件管理系统范围以及专用连接配置集

默认情况下, 主机上的所有用户都可以使用 `ifcfg` 文件中定义的连接。您可以通过在 `ifcfg` 文件中添加 `USERS` 参数, 来将此行为限制为特定用户。

#### 先决条件

- `ifcfg` 文件已存在。

#### 流程

1.

编辑 `/etc/sysconfig/network-scripts/` 目录中您要限制为某些用户的 `ifcfg` 文件, 并添加 :

```
USERS="username1 username2 ..."
```

2.

重新激活连接 :

```
nmcli connection up connection_name
```

## 第 38 章 KEYFILE 格式的 NETWORKMANAGER 连接配置文件

默认情况下，NetworkManager 以 ifcfg 格式存储连接配置文件，但您也可以使用 keyfile 格式的配置文件。与已弃用的 ifcfg 格式不同，keyfile 格式支持 NetworkManager 提供的所有连接设置。

在 Red Hat Enterprise Linux 9 中，keyfile 格式将是默认值。

### 38.1. NETWORKMANAGER 配置文件的密钥文件格式

keyfile 格式与 INI 格式类似。例如，以下是 keyfile 格式的以太网连接配置文件：

```
[connection]
id=example_connection
uuid=82c6272d-1ff7-4d56-9c7c-0eb27c300029
type=ethernet
autoconnect=true

[ipv4]
method=auto

[ipv6]
method=auto

[ethernet]
mac-address=00:53:00:8f:fa:66
```



#### 警告

参数的拼写错误或不正确的放置可能会导致意外行为。因此，请不要手动编辑或创建 NetworkManager 配置文件。

使用 nmcli 工具、network RHEL 系统角色或 nmstate API 来管理 NetworkManager 连接。例如，您可以 [在离线模式下使用 nmcli 工具 创建连接配置文件](#)。

每个部分都对应一个 NetworkManager 设置名称，如 [nm-settings \(5\)](#) 手册页中所述。该部分中的每

一个键值对是手册页设置规范中列出的一个属性。

**NetworkManager** 密钥文件中的大部分变量都具有一对一映射。这意味着 **NetworkManager** 属性以相同的名称和格式的变量存储在密钥文件中。不过，也有例外情况，主要是为了使密钥文件语法更易于阅读。有关这些例外的列表，请查看您系统上的 **nm-settings-keyfile (5)** 手册页。



### 重要

为安全起见，因为连接配置文件可以包含敏感信息，如私钥和密码短语，所以 **NetworkManager** 仅使用 **root** 用户所拥有的配置文件，并且仅可由 **root** 读写。

将带有.nmconnection 后缀的连接配置文件保存在 `/etc/NetworkManager/system-connections/` 目录中。此目录包含持久性配置文件。如果您使用 **NetworkManager API** 修改了持久配置文件，**NetworkManager** 会写并覆盖此目录中的文件。

**NetworkManager** 不会自动从磁盘重新加载配置文件。当您以密钥文件格式创建或更新连接配置文件时，请使用 `nmcli connection reload` 命令告知 **NetworkManager** 这些变化。

## 38.2. 使用 NMCLI 在离线模式下创建 KEYFILE 连接配置文件

使用 **NetworkManager** 工具（如 `nmcli`）、**network RHEL** 系统角色或 `nmstate API` 来管理 **NetworkManager** 连接，以创建和更新配置文件。但是，您也可以使用 `nmcli --offline connection add` 命令，在离线模式下创建各种 `keyfile` 格式的连接配置文件。

脱机模式可确保 `nmcli` 在没有 **NetworkManager** 服务的情况下运行，以通过标准输出生成 `keyfile` 连接配置集。此功能在以下场景下很有用：

- 您需要创建需要预先部署的连接配置集。例如在容器镜像中，或者作为 **RPM** 软件包。
- 您希望在 **NetworkManager** 服务不可用的环境中创建连接配置文件，例如当您要使用 `chroot` 工具时。或者，当您想通过 `Kickstart %post` 脚本创建或修改 **RHEL** 系统的网络配置时。

### 流程

- 1.

以 keyfile 格式创建新连接配置集。例如，对于不使用 DHCP 的以太网设备的连接配置文件，请运行类似的 nmcli 命令：

```
nmcli --offline connection add type ethernet con-name Example-Connection
 ipv4.addresses 192.0.2.1/24 ipv4.dns 192.0.2.200 ipv4.method manual >
 /etc/NetworkManager/system-connections/example.nmconnection
```



### 注意

使用 `con-name` 键指定的连接名称被保存到生成的配置集的 `id` 变量中。当您使用 nmcli 命令稍后管理这个连接时，请按如下所示指定连接：

- 如果没有省略 `id` 变量，请使用连接名称，如 `Example-Connection`。
- 当没有使用 `id` 变量时，请使用没有 `.nmconnection` 后缀的文件名，如 `output`。

2.

对配置文件设置权限，以便只有 root 用户可以读和更新它：

```
chmod 600 /etc/NetworkManager/system-connections/example.nmconnection
chown root:root /etc/NetworkManager/system-connections/example.nmconnection
```

3.

启动 NetworkManager 服务：

```
systemctl start NetworkManager.service
```

4.

如果将配置文件中的 `autoconnect` 变量设置为 `false`，则激活连接：

```
nmcli connection up Example-Connection
```

### 验证

1.

验证 NetworkManager 服务是否正在运行：

```
systemctl status NetworkManager.service
● NetworkManager.service - Network Manager
```

```

 Loaded: loaded (/usr/lib/systemd/system/NetworkManager.service; enabled; vendor
 preset: enabled)
 Active: active (running) since Wed 2022-08-03 13:08:32 CEST; 1min 40s ago
 ...

```

2.

验证 NetworkManager 是否可以从配置文件中读取配置集：

```

nmcli -f TYPE,FILENAME,NAME connection
 NAME
TYPE FILENAME
ethernet /etc/NetworkManager/system-connections/examaple.nmconnection Example-
Connection
ethernet /etc/sysconfig/network-scripts/ifcfg-enp1s0 enp1s0
...

```

如果输出没有显示新创建的连接，请验证密钥文件权限和您所用的语法是否正确。

3.

显示连接配置文件：

```

nmcli connection show Example-Connection
connection.id: Example-Connection
connection.uuid: 232290ce-5225-422a-9228-cb83b22056b4
connection.stable-id: --
connection.type: 802-3-ethernet
connection.interface-name: --
connection.autoconnect: yes
...

```

## 其他资源



您系统上的 **nmcli** (1)、**nm-settings** (5) 和 **nm-settings-keyfile** (5) 手册页

### 38.3. 手动创建 KEYFILE 格式的 NETWORKMANAGER 配置文件

您可以手动创建密钥文件格式的 NetworkManager 连接配置集。



## 警告

手动创建或更新配置文件可能会导致意外或无法正常工作的网络配置。作为备用方案，您可以在离线模式下使用 nmcli。请参阅 [使用 nmcli 在离线模式下创建 keyfile 连接配置文件](#)

## 流程

1.

创建连接配置文件。例如，对于使用 DHCP 的 enp1s0 以太网设备的连接配置文件，请创建具有以下内容的 /etc/NetworkManager/system-connections/example.nmconnection 文件：

```
[connection]
id=Example-Connection
type=ethernet
autoconnect=true
interface-name=enp1s0

[ipv4]
method=auto

[ipv6]
method=auto
```



## 注意

您可以使用任何以 .nmconnection 为后缀的文件名。但是，当您稍后使用 nmcli 命令来管理连接时，您必须在引用此连接时使用 id 变量中设置的连接名称。当省略 id 变量时，请使用不带 .nmconnection 的文件名来引用此连接。

2.

对配置文件设置权限，以便只有 root 用户可以读和更新它：

```
chown root:root /etc/NetworkManager/system-connections/example.nmconnection
chmod 600 /etc/NetworkManager/system-connections/example.nmconnection
```

3.

重新加载连接配置文件：

```
nmcli connection reload
```

4.

验证 NetworkManager 是否从配置文件读取配置文件：

```
nmcli -f NAME,UUID,FILENAME connection
NAME UUID FILENAME
Example-Connection 86da2486-068d-4d05-9ac7-957ec118afba
/etc/NetworkManager/system-connections/example.nmconnection
...
```

如果命令未显示新添加的连接，请验证文件权限和您在文件中使用的语法是否正确。

5.

如果将配置文件中的 `autoconnect` 变量设置为 `false`，则激活连接：

```
nmcli connection up example_connection
```

验证



显示连接配置文件：

```
nmcli connection show example_connection
```

其他资源



您系统上的 `nm-settings (5)` 和 `nm-settings-keyfile (5)` 手册页

#### 38.4. IFCFG 和 KEYFILE 格式的配置文件在接口重命名方面的差异

您可以定义自定义网络接口名称，如 `provider` 或 `lan`，使接口名称更具描述性。在这种情况下，`udev` 服务会重命名接口。重命名过程会因您使用 `ifcfg` 或 `keyfile` 格式的连接配置文件而有所不同。

使用 `ifcfg` 格式的配置文件时的接口重命名过程

1.

`udev` 规则文件 `/usr/lib/udev/rules.d/60-net.rules` 调用 `/lib/udev/rename_device helper` 工具。

2.

助手工具在 `/etc/sysconfig/network-scripts/ifcfg-*` 文件中搜索 `HWADDR` 参数。

3.

如果变量中设置的值与接口的 MAC 地址匹配，则帮助工具会将接口重命名为文件的 **DEVICE** 参数中设置的名称。

### 使用 keyfile 格式的配置文件时的接口重命名过程

1.

创建一个 [systemd 链接文件](#) 或 [udev 规则](#) 来重命名接口。

2.

在 NetworkManager 连接配置文件的 **interface-name** 属性中使用自定义接口名称。

### 其他资源



[udev 设备管理器如何重命名网络接口](#)



[使用 udev 规则配置用户定义的网络接口名称](#)



[使用 systemd 链接文件配置用户定义的网络接口名称](#)

### 38.5. 将 NETWORKMANAGER 配置集从 IFCFG 迁移到 KEYFILE 格式

如果使用 ifcfg 格式的连接配置文件，您可以将它们转换为 keyfile 格式，使所有配置文件都具有首选的格式，并位于一个位置。



#### 注意

如果 ifcfg 文件包含 **NM\_CONTROLLED=no** 设置，NetworkManager 不会控制这个配置文件，因此迁移过程会忽略它。

### 先决条件



您有 ifcfg 格式的连接配置集，采用 `/etc/sysconfig/network-scripts/` 目录中。



如果连接配置文件包含一个被设置为自定义设备名称的 **DEVICE** 变量，如 `provider` 或 `lan`，

那么您为每个自定义设备名称创建了一个 [systemd 链接文件](#) 或 [udev 规则](#)。

## 流程

- 迁移连接配置集：

```
nmcli connection migrate
Connection 'enp1s0' (43ed18ab-f0c4-4934-af3d-2b3333948e45) successfully migrated.
Connection 'enp2s0' (883333e8-1b87-4947-8ceb-1f8812a80a9b) successfully migrated.
...
```

## 验证

- 另外，您可以验证您是否成功迁移了所有连接配置集：

```
nmcli -f TYPE,FILENAME,NAME connection
 TYPE FILENAME NAME
ethernet /etc/NetworkManager/system-connections/enp1s0.nmconnection enp1s0
ethernet /etc/NetworkManager/system-connections/enp2s0.nmconnection enp2s0
...
```

## 其他资源

- [nm-settings-keyfile\(5\)](#)
- [nm-settings-ifcfg-rh\(5\)](#)
- [udev 设备管理器如何重命名网络接口](#)

## 第 39 章 SYSTEMD 网络目标和服务

在应用网络设置时，RHEL 使用 `network` 和 `network-online` 目标以及 `NetworkManager-wait-online` 服务。另外，如果这些服务希望网络启动，且它们无法动态响应网络状态的变化，您可以将 `systemd` 服务配置为在网络完全可用后启动。

### 39.1. 网络和网络在线 SYSTEMD 目标的区别

`systemd` 维护 `network` 和 `network-online` 目标单元。特殊单元，如 `NetworkManager-wait-online.service`，具有 `WantedBy=network-online.target` 和 `Before=network-online.target` 参数。如果启用了，这些单元将启动 `network-online.target`，并延迟要达到的目标，直到建立了某种形式的网络连接。它们会延迟 `network-online` 目标，直到网络连接了。

`network-online` 目标启动一个服务，这会对进一步执行增加更长的延迟。`systemd` 会自动使用这个目标单元的 `Wants` 和 `After` 参数来向所有 System V(SysV) init 脚本服务单元添加依赖项，这些服务单元具有一个指向 `$network` 工具的 Linux Standard Base(LSB)头。LSB 头是 init 脚本的元数据。您可以使它指定依赖项。这与 `systemd` 目标类似。

`network` 目标不会显著延迟引导进程的执行。到达 `network` 目标意味着，负责设置网络的服务已启动。但并不意味着已经配置了一个网络设备。这个目标在关闭系统的过程中非常重要。例如，如果您在引导过程中有一个排在 `network` 目标之后的服务，则这个依赖关系在关闭过程中会反过来。在服务停止后，网络才会断开连接。远程网络文件系统的所有挂载单元都会自动启动 `network-online` 目标单元，并在其之后排序。



#### 注意

`network-online` 目标单元仅在系统启动期间有用。系统完成引导后，这个目标不会跟踪网络的在线状态。因此，您无法使用 `network-online` 来监控网络连接。这个目标提供一个一次性系统启动概念。

### 39.2. NETWORKMANAGER-WAIT-ONLINE 概述

`NetworkManager-wait-online` 服务延迟到达 `network-online` 目标，直到 `NetworkManager` 报告启动已完成。在启动过程中，`NetworkManager` 会通过将 `connection.autoconnect` 参数设置为 `yes` 来激活所有配置文件。但是，只要 `NetworkManager` 配置集处于激活状态，配置文件的激活就没有完成。如果激活失败，`NetworkManager` 会根据 `connection.autoconnect-retries` 的值来重试激活。

设备何时达到激活状态取决于其配置。例如，如果配置文件同时包含 IPv4 和 IPv6 配置，默认情况

下, NetworkManager 会在只有一个地址系列就绪时将设备视为完全激活。连接配置文件中的 `ipv4.may-fail` 和 `ipv6.may-fail` 参数控制此行为。

对于以太网设备, NetworkManager 会超时等待载体。因此, 如果以太网电缆没有连接, 则这可能会进一步延迟 `NetworkManager-wait-online.service`。

当启动完成后, 所有配置集都处于断开连接的状态, 或被成功激活。您可以配置配置集来自动连接。以下是一些参数示例, 这些参数设定超时或者在连接被视为活跃时定义:

- `connection.wait-device-timeout` : 设置驱动程序检测设备的超时时间。
- `ipv4.may-fail` 和 `ipv6.may-fail` : 使用一个就绪的 IP 地址系列设置激活, 或者一个特定的地址系列是否已完成配置。
- `ipv4.gateway-ping-timeout`: Delays 网络激活, 直到 NetworkManager 接收来自 IPv4 网关的 ping 响应。在继续操作前, 系统会最多等待指定秒数。

## 其他资源

- 您系统上的 `nm-settings` (5), `systemd.special` (7), `NetworkManager-wait-online.service` (8) 手册页

### 39.3. 将 SYSTEMD 服务配置为在网络已启动后再启动

Red Hat Enterprise Linux 在 `/usr/lib/systemd/system/` 目录中安装 systemd 服务文件。此流程为 `/etc/systemd/system/<service_name>.service.d/` 中的一个服务文件创建一个置入片断, 其与 `/usr/lib/systemd/system/` 中的服务文件一起使用, 以便在网络在线后启动特定的服务。如果置入段中的设置与 `/usr/lib/systemd/system/` 中服务文件中的设置重叠, 则它具有更高的优先级。

## 流程

1. 在编辑器中打开服务文件 :

```
systemctl edit <service_name>
```

2.

输入以下内容并保存更改：

```
[Unit]
After=network-online.target
```

3.

重新加载 **systemd** 服务。

```
systemctl daemon-reload
```

## 第 40 章 NMSTATE 简介

**nmstate** 是一个声明性网络管理器 API。使用 **Nmstate** 时，您可以使用 YAML 或 JSON 格式的指令描述预期的网络状态。

**nmstate** 有很多优点。例如，它：

- 提供稳定且可扩展的接口来管理 RHEL 网络功能
- 支持主机和集群级别的原子和事务操作
- 支持对大多数属性进行部分编辑，并保留在说明中没有指定的现有设置
- 提供插件支持，使管理员能够使用自己的插件

**Nmstate** 由以下软件包组成：

软件包	内容
<b>nmstate</b>	<b>nmstatectl</b> 命令行工具
<b>python3-libnmstate</b>	<b>libnmstate</b> Python 库
<b>nmstate-libs</b>	Nmstate C 库
<b>nmstate-devel</b>	Nmstate C 库标头

### 40.1. 在 PYTHON 应用程序中使用 LIBNMSTATE 库

**libnmstate** Python 库可让开发人员在他们自己的应用程序中使用 **Nmstate**

要使用库，请在源代码中导入它：

```
import libnmstate
```

请注意，您必须安装 `nmstate` 和 `python3-libnmstate` 软件包才能使用这个库。

#### 例 40.1. 使用 `libnmstate` 库查询网络状态

以下 Python 代码导入了 `libnmstate` 库，并显示可用的网络接口及其状态：

```
import libnmstate
from libnmstate.schema import Interface

net_state = libnmstate.show()
for iface_state in net_state[Interface.KEY]:
 print(iface_state[Interface.NAME] + ":" +
 iface_state[Interface.STATE])
```

#### 40.2. 使用 `NMSTATECTL` 更新当前网络配置

您可以使用 `nmstatectl` 工具将一个或多个接口的当前网络配置存储在一个文件中。然后您可以使用此文件：

- 修改配置并将其应用到同一系统。
- 将文件复制到其他主机上，并使用相同的或经过修改的设置配置主机。

例如，您可以将 `enp1s0` 接口的设置导出到一个文件中，修改配置，将设置应用到主机。

##### 前提条件

- `nmstate` 软件包已安装。

##### 流程

1. 将 `enp1s0` 接口的设置导出到 `~/network-config.yml` 文件：

```
nmstatectl show enp1s0 > ~/network-config.yml
```

此命令会以 YAML 格式存储 `enp1s0` 的配置。要以 JSON 格式存储输出，请将 `--json` 选项传给命令。

如果没有指定接口名称，`nmstatectl` 将导出所有接口的配置。

2. 使用文本编辑器修改 `~/network-config.yml` 文件，以更新配置。

3. 应用 `~/network-config.yml` 文件中的设置：

```
nmstatectl apply ~/network-config.yml
```

如果您以 JSON 格式导出设置，请将 `--json` 选项传给命令。

#### 40.3. NETWORK RHEL 系统角色的网络状态

`network` RHEL 系统角色支持 `playbook` 中的状态配置来配置设备。为此，请使用 `network_state` 变量，后面跟上状态配置。

在 `playbook` 中使用 `network_state` 变量的好处：

- 通过与状态配置结合使用声明方法，您可以配置接口，`NetworkManager` 会在后台为这些接口创建一个配置集。
- 使用 `network_state` 变量，您可以指定您需要更改的选项，所有其他选项将保持不变。但是，使用 `network_connections` 变量，您必须指定所有设置来更改网络连接配置集。



#### 重要

您只能在 `network_state` 设置 `Nmstate` YAML 指令。这些指令与您可以在 `network_connections` 中设置的变量有所不同。

例如，要使用动态 IP 地址设置创建以太网连接，请在 playbook 中使用以下 vars 块：

带有状态配置的 playbook	常规 playbook
<pre>vars:   network_state:     interfaces:       - name: enp7s0         type: ethernet         state: up         ipv4:           enabled: true           auto-dns: true           auto-gateway: true           auto-routes: true           dhcp: true         ipv6:           enabled: true           auto-dns: true           auto-gateway: true           auto-routes: true           autoconf: true           dhcp: true</pre>	<pre>vars:   network_connections:     - name: enp7s0       interface_name: enp7s0       type: ethernet       autoconnect: yes       ip:         dhcp4: yes         auto6: yes       state: up</pre>

例如，要仅更改您之前创建的动态 IP 地址设置的连接状态，请在 playbook 中使用以下 vars 块：

带有状态配置的 playbook	常规 playbook
<pre>vars:   network_state:     interfaces:       - name: enp7s0         type: ethernet         state: down</pre>	<pre>vars:   network_connections:     - name: enp7s0       interface_name: enp7s0       type: ethernet       autoconnect: yes       ip:         dhcp4: yes         auto6: yes       state: down</pre>

## 其他资源

- 

</usr/share/ansible/roles/rhel-system-roles.network/README.md> 文件

- **/usr/share/doc/rhel-system-roles/network/ 目录**

## 第 41 章 使用和配置 FIREWALLD

防火墙是保护机器不受来自外部的、不需要的网络数据影响的一种方式。它允许用户通过定义一组防火墙规则来控制主机上的入站网络流量。这些规则用于对传入流量进行排序，并阻止它或允许它通过。

**firewalld** 是一个防火墙服务守护进程，它通过 D-Bus 接口提供动态、可自定义的防火墙。如果是动态的，它会在每次更改规则时启用、修改和删除规则，而无需在每次更改规则时重启防火墙守护进程。

您可以使用 **firewalld** 来配置大多数典型情况所需的数据包过滤。如果 **firewalld** 不涵盖您的场景，或者您想要完全控制规则，请使用 **nftables** 框架。

**firewalld** 使用区、策略和服务的概念来简化流量管理。区域以逻辑方式分隔网络。网络接口和源可以分配给区。策略用于拒绝或允许区域间的流量流。防火墙服务是预定义的规则，覆盖了允许特定服务的传入流量的所有必要设置，并在区域内应用。

服务使用一个或多个端口或地址进行网络通信。防火墙会根据端口过滤通讯。要允许服务的网络流量，必须打开其端口。**firewalld** 会阻止未明确设置为开放的端口上的所有流量。一些区（如可信区）默认允许所有流量。

**firewalld** 维护单独的运行时和永久配置。这允许仅运行时的更改。**firewalld** 重新加载或重启后，运行时配置不会保留。在启动时，它会填充自永久配置。

请注意，带有 **nftables** 后端的 **firewalld** 不支持使用 **-direct** 选项将自定义 **nftables** 规则传递给 **firewalld**。

### 41.1. 使用 FIREWALLD、NFTABLES 或者 IPTABLES 时

在 RHEL 8 中，您可以根据您的情况使用以下数据包过滤工具：

- **firewalld** : **firewalld** 工具简化了常见用例的防火墙配置。
- **nftables** : 使用 **nftables** 工具来设置复杂和性能关键的防火墙，如用于整个网络。

- **iptables** : Red Hat Enterprise Linux 上的 **iptables** 工具使用 **nf\_tables** 内核 API 而不是传统的后端。**nf\_tables** API 提供了向后兼容性，以便使用 **iptables** 命令的脚本仍可在 Red Hat Enterprise Linux 上工作。对于新的防火墙脚本，请使用 **nftables**。



### 重要

要防止不同的与防火墙相关的服务(firewalld、nftables 或 iptables)相互影响，请在 RHEL 主机上仅运行其中一个服务，并禁用其他服务。

## 41.2. 防火墙区域

您可以使用 **firewalld** 工具，根据您与网络中接口和流量的信任级别，将网络划分为不同的区域。连接只能是一个区域的一部分，但您可以对许多网络连接使用这个区域。

**firewalld** 遵循严格的原则：

1. 流量只进入一个区域。
2. 流量只流出一个区域。
3. 一个区域定义了一个信任级别。
4. 默认情况下，允许区域内流量（在同一区域中）。
5. 默认情况下，拒绝区域间流量（从区域到区域）。

原则 4 和 5 是原则 3 的结果。

原则 4 可以通过区选项 **--remove-forward** 进行配置。原则 5 可以通过添加新策略来进行配置。

NetworkManager 通知接口区的 firewalld。您可以使用以下工具将区域分配给接口：

- **NetworkManager**
- **firewall-config 工具**
- **firewall-cmd 工具**
- **RHEL web 控制台**

RHEL web 控制台、firewall-config 和 firewall-cmd 只能编辑合适的 NetworkManager 配置文件。如果您使用 web 控制台、firewall-cmd 或 firewall-config 更改接口的区域，请求被转发到 NetworkManager，且不会被 firewalld 处理。

/usr/lib/firewalld/zones/ 目录存储预定义的区域，您可以立即将它们应用到任何可用的网络接口。只有在修改后，这些文件才会被拷贝到 /etc/firewalld/zones/ 目录中。预定义区的默认设置如下：

#### **block**

- 适合于：任何传入的网络连接都会被拒绝，对于 IPv4 显示 icmp-host-prohibited 消息，对于 IPv6 显示 icmp6-adm-prohibited 消息。
- 接受：只有从系统中启动的网络连接。

#### **dmz**

- 适用于：DMZ 中的计算机，可使用对内部网络的有限权限访问它们。
- 接受：只有所选的传入连接。

#### **drop**

适用于：任何传入的网络数据包都被丢弃，没有任何通知。

- 接受：仅传出的网络连接。

#### **external**

- 适用于：启用了伪装的外部网络，尤其适用于路由器。不信任网络上其他计算机的情况。
- 接受：只有所选的传入连接。

#### **home**

- 适用于：您主要信任网络上其他计算机的家庭环境。
- 接受：只有所选的传入连接。

#### **internal**

- 适用于：您主要信任网络上其他计算机的内部网络。
- 接受：只有所选的传入连接。

#### **public**

- 适用于：您不信任网络上其他计算机的公共区域。
- 接受：只有所选的传入连接。

#### **trusted**

- 接受：所有网络连接。

## work

适用于：您主要信任网络上其他计算机的工作环境。

- 接受：只有所选的传入连接。

这些区中的一个被设置为 **default** 区。当接口连接被添加到 NetworkManager 中时，它们会被分配到默认区。安装时，firewalld 中的默认区域是 **public** 区域。您可以更改默认区域。



### 注意

使网络区域名称自我解释，以帮助用户快速理解它们。

要避免安全问题，请查看默认区配置并根据您的需要和风险禁用任何不必要的服务。

## 其他资源

- 您系统上的 **firewalld.zone (5)** 手册页

### 41.3. 防火墙策略

防火墙策略指定网络所需的安全状态。它们概述了对不同类型的流量所采取的规则和操作。通常，策略包含用于以下类型流量的规则：

- 传入流量
- 传出流量
- 转发流量

- 特定服务和应用程序
- 网络地址转换(NAT)

防火墙策略使用防火墙区域的概念。每个区域都与一组决定允许的流量的特定的防火墙规则关联。策略以有状态、单向的方式应用防火墙规则。这意味着您只考虑流量的一个方向。由于 `firewalld` 的有状态过滤，流量返回路径被隐式允许。

策略与一个入口区域和一个出口区域关联。入口区域是流量起源的地方（接收）。出口区域是流量离开的地方(发送)。

策略中定义的防火墙规则可以引用防火墙区，以便在多个网络接口之间应用一致的配置。

#### 41.4. 防火墙规则

您可以使用防火墙规则实现特定的配置，以允许或阻止网络流量。因此，您可以控制网络流量的流，以防止系统受到安全威胁。

防火墙规则通常根据各种属性定义某些条件。属性可以是如下：

- 源 IP 地址
- 目标 IP 地址
- 传输协议(TCP、 UDP、 ...)
- 端口
- 网络接口

**firewalld** 工具将防火墙规则组织到区域(如public、internal 等)和策略中。每个区域都有自己的一组规则，其决定与特定区域关联的网络接口的流量自由度的级别。

#### 41.5. 防火墙直接规则

**firewalld** 服务提供多种配置规则的方法，包括：

- 常规规则
- 直接规则

这两者的一个区别在于，每个方法与底层后端(**iptables** 或 **nftables**)交互的方式。

直接规则是高级低级别规则，允许直接与 **iptables** 交互。它们绕过 **firewalld** 的结构化管理，为您提供更多控制。您可以使用原始 **iptables** 语法使用 **firewall-cmd** 命令手动定义直接规则。例如，**firewall-cmd --direct --add-rule ipv4 filter INPUT 0 -s 198.51.100.1 -j DROP**。此命令添加了一个 **iptables** 规则来丢弃来自 198.51.100.1 源 IP 地址的流量。

但是，使用直接规则也具有其缺点。特别是当 **nftables** 是您的主要防火墙后端时。例如：

- 直接规则很难维护，并可能会与基于 **nftables** 的 **firewalld** 配置冲突。
- 直接规则不支持您可以在 **nftables** 中找到的高级功能，如原始表达式和有状态对象。
- 直接规则不永存。**iptables** 组件已弃用，最终将从 RHEL 中删除。

因此，您可以考虑使用 **nftables** 替换 **firewalld** 直接规则。查看知识库解决方案 [如何将 firewalld 直接规则替换为 nftables？](#) 以查看更多详情。

#### 41.6. 预定义的 FIREWALLD 服务

预定义的 firewalld 服务在低级防火墙规则中提供内置抽象层。它通过将常用的网络服务（如 SSH 或 HTTP）映射到其相应的端口和协议来实现。您可以引用指定预定义服务，而不是每次手动指定它们。这使得防火墙管理变得更加简单、更易出错且更直观。

- 

查看可用的预定义服务：

```
firewall-cmd --get-services
RH-Satellite-6 RH-Satellite-6-capsule afp amanda-client amanda-k5-client amqp amqps
apcupsd audit ausweisapp2 bacula bacula-client bareos-director bareos-filedaemon
bareos-storage bb bgp bitcoin bitcoin-rpc bitcoin-testnet bitcoin-testnet-rpc bittorrent-
lsd ceph ceph-exporter ceph-mon cfengine checkmk-agent cockpit collectd condor-
collector cratedb ctdb dds...
```

- 

要进一步检查特定的预定义服务：

```
sudo firewall-cmd --info-service=RH-Satellite-6
RH-Satellite-6
 ports: 5000/tcp 5646-5647/tcp 5671/tcp 8000/tcp 8080/tcp 9090/tcp
 protocols:
 source-ports:
 modules:
 destination:
 includes: foreman
 helpers:
```

示例输出显示 RH-Satellite-6 预定义服务侦听端口 5000/tcp 5646-5647/tcp 5671/tcp 8000/tcp 8080/tcp 9090/tcp 9090/tcp 9090。另外，RH-Satellite-6 继承了其他预定义服务中的规则。本例中为 foreman。

每个预定义的服务都作为 XML 文件存储在 /usr/lib/firewalld/services/ 目录中。

## 其他资源

- 

[firewall-cmd \(1\), firewalld \(1\) 手册页](#)

### 41.7. 使用 FIREWALLD 区

**zones** 代表一种更透明管理传入流量的概念。这些区域连接到联网接口或者分配一系列源地址。您可以独立为每个区管理防火墙规则，这样就可以定义复杂的防火墙设置并将其应用到流量。

#### 41.7.1. 自定义特定区域的防火墙设置，以增强安全性

您可以通过修改防火墙设置并将特定的网络接口或连接与特定的防火墙区域关联，来增强网络安全。通过为区域定义细粒度规则和限制，您可以根据想要的安全级别控制入站和出站流量。

例如，您可以获得以下好处：

- 保护敏感数据
- 防止未授权访问
- 缓解潜在的网络威胁

#### 前提条件

- firewalld 服务正在运行。

#### 流程

1. 列出可用的防火墙区域：

```
firewall-cmd --get-zones
```

`firewall-cmd --get-zones` 命令显示系统上所有可用的区，但不显示特定区的详情。要查看所有区域的详情，请使用 `firewall-cmd --list-all-zones` 命令。

2. 选择您要用于此配置的区域。
3. 修改所选区域的防火墙设置。例如，要允许 SSH 服务，并删除 ftp 服务：

```
firewall-cmd --add-service=ssh --zone=<your_chosen_zone>
firewall-cmd --remove-service=ftp --zone=<same_chosen_zone>
```

4.

将一个网络接口分配给防火墙区域：

a.

列出可用的网络接口：

```
firewall-cmd --get-active-zones
```

区域的活动是由存在的网络接口或与其配置匹配的源地址范围确定的。默认区域对于未分类的流量处于活跃状态，但如果 没有流量匹配其规则，则始终处于活跃状态。

b.

将一个网络接口分配给所选区域：

```
firewall-cmd --zone=<your_chosen_zone> --change-interface=<interface_name> -permanent
```

将一个网络接口分配给一个区域更适合将一致的防火墙设置应用到特定接口（物理或虚拟）上的所有流量。

当 `firewall-cmd` 命令与 `--permanent` 选项一起使用时，通常涉及更新 `NetworkManager` 连接配置文件，以永久更改防火墙配置。`firewalld` 和 `NetworkManager` 之间的这种集成确保一致的网络和防火墙设置。

## 验证

1.

显示选择区域的更新设置：

```
firewall-cmd --zone=<your_chosen_zone> --list-all
```

命令输出显示所有区域设置，包括分配的服务、网络接口和网络连接（源）。

### 41.7.2. 更改默认区

系统管理员在其配置文件中为网络接口分配区域。如果接口没有被分配给指定区，它将被分配给默认区。每次重启 `firewalld` 服务后，`firewalld` 会加载默认区的设置，并使其处于活动状态。请注意，所有其他区域的设置都被保留，并随时可以使用。

通常，NetworkManager 根据 NetworkManager 连接配置文件中的 `connection.zone` 设置将区域分配给接口。另外，重启后，NetworkManager 管理“激活”这些区域的分配。

## 前提条件

- `firewalld` 服务正在运行。

## 流程

设置默认区：

1. 显示当前的默认区：

```
firewall-cmd --get-default-zone
```

2. 设置新的默认区：

```
firewall-cmd --set-default-zone <zone_name>
```



注意

按照此流程，设置是一个永久设置，即使没有 `--permanent` 选项。

### 41.7.3. 将网络接口分配给区

可以为不同区定义不同的规则集，然后通过更改所使用的接口的区来快速改变设置。使用多个接口，可以为每个具体区设置一个区来区分通过它们的网络流量。

## 流程

要将区分配给特定的接口：

1. 列出活跃区以及分配给它们的接口：

```
firewall-cmd --get-active-zones
```

2.

为不同的区分配接口：

```
firewall-cmd --zone=zone_name --change-interface=interface_name --permanent
```

#### 41.7.4. 添加源

要将传入的流量路由到特定区，请将源添加到那个区。源可以是一个使用 CIDR 格式的 IP 地址或 IP 掩码。



注意

如果您添加多个带有重叠网络范围的区域，则根据区名称排序，且只考虑第一个区。

●

在当前区中设置源：

```
firewall-cmd --add-source=<source>
```

●

要为特定区设置源 IP 地址：

```
firewall-cmd --zone=zone-name --add-source=<source>
```

以下流程允许来自 受信任 区中 192.168.2.15 的所有传入的流量：

#### 流程

1.

列出所有可用区：

```
firewall-cmd --get-zones
```

2.

将源 IP 添加到持久性模式的信任区中：

```
firewall-cmd --zone=trusted --add-source=192.168.2.15
```

3.

使新设置具有持久性：

```
firewall-cmd --runtime-to-permanent
```

#### 41.7.5. 删除源

当您从区域删除源时，源自源的流量不再被通过该源指定的规则定向。相反，流量会返回到与它源自的接口关联的区域的规则和设置，或进到默认区域。

#### 流程

1.

列出所需区的允许源：

```
firewall-cmd --zone=zone-name --list-sources
```

2.

从区永久删除源：

```
firewall-cmd --zone=zone-name --remove-source=<source>
```

3.

使新设置具有持久性：

```
firewall-cmd --runtime-to-permanent
```

#### 41.7.6. 使用 nmcli 为连接分配区域

您可以使用 nmcli 实用程序在 NetworkManager 连接中添加 firewalld 区域。

#### 流程

1.

将区分配给 NetworkManager 连接配置文件：

```
nmcli connection modify profile connection.zone zone_name
```

2.

激活连接：

```
nmcli connection up profile
```

#### 41.7.7. 在 ifcfg 文件中手动将区分配给网络连接

当连接由 NetworkManager 管理时，必须了解它使用的区。对于每个网络连接配置文件，可以指定一个区域，其根据带有可移植设备的计算机的位置提供各种防火墙设置的灵活性。因此，可以为不同的位置（如公司或家）指定区域和设置。

#### 流程

- 要为连接设置一个区，请编辑 `/etc/sysconfig/network-scripts/ifcfg-connection_name` 文件，并添加将区分配给这个连接的行：

```
ZONE=zone_name
```

#### 41.7.8. 创建一个新区

要使用自定义区，创建一个新的区并使用它像预定义区一样。新区需要 `--permanent` 选项，否则命令无法工作。

#### 先决条件

- `firewalld` 服务在运行。

#### 流程

1. 创建一个新区：

```
firewall-cmd --permanent --new-zone=zone-name
```

2. 使新区域可用：

```
firewall-cmd --reload
```

命令将最新的更改应用到防火墙配置，而不中断已在运行的网络服务。

#### 验证

- 检查是否在您的永久设置中添加了新的区：

```
firewall-cmd --get-zones --permanent
```

#### 41.7.9. 使用 Web 控制台启用区域

您可以通过 RHEL web 控制台对特定接口或 IP 地址范围应用预定义和现有的防火墙区域。

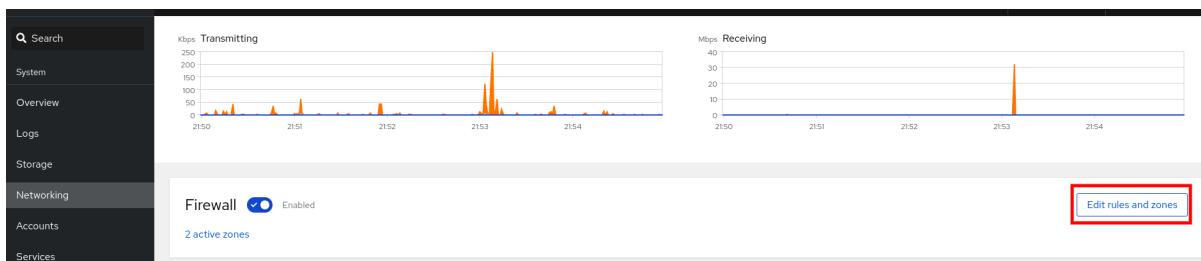
##### 先决条件

- 您已安装了 RHEL 8 web 控制台。
- 您已启用了 cockpit 服务。
- 您的用户帐户被允许登录到 web 控制台。

具体步骤请参阅[安装并启用 Web 控制台](#)。

##### 流程

1. 登录到 RHEL 8 web 控制台。  
详情请参阅[登录到 web 控制台](#)。
2. 点 Networking。
3. 点编辑规则和区域按钮。



如果没有看到 **Edit rules and zones** 按钮，使用管理员权限登录到 web 控制台。

4. 在 **Firewall** 部分，点 **Add new zone**。
5. 在 **Add zone** 对话框中，从信任级别选项选择一个区。  
Web 控制台显示 firewalld 服务中预定义的所有区域。
6. 在接口部分，选择一个应用所选区的接口或接口。  
在 **Allowed Addresses** 部分中，您可以选择是否应用该区：
  - 整个子网
  - 或者以以下格式表示的 IP 地址范围：
    - **192.168.1.0**
    - **192.168.1.0/24**
    - **192.168.1.0/24, 192.168.1.0**

8.

点 **Add zone** 按钮。

**Add zone**

<b>Trust level</b>	Sorted from least to most trusted	Custom zones
<input type="radio"/> Public <input type="radio"/> External <input type="radio"/> Dmz <input type="radio"/> Work <input checked="" type="radio"/> Home <input type="radio"/> Internal	<input type="radio"/> FedoraServer	
<b>Description</b>		
For use in home areas. You mostly trust the other computers on networks to not harm your computer. Only selected incoming connections are accepted.		
<b>Included services</b>		
ssh, mdns, samba-client, dhcpcv6-client The cockpit service is automatically included		
<b>Interfaces</b>		
<input type="checkbox"/> enp0s20f0u4u1u2 <input checked="" type="checkbox"/> enp0s31f6 <input type="checkbox"/> p2p-dev-wlp6s0 <input type="checkbox"/> tap0 <input type="checkbox"/> tun0		
<b>Allowed addresses</b>		
<input checked="" type="radio"/> Entire subnet <input type="radio"/> Range		
<input style="background-color: #0070C0; color: white; border-radius: 5px; padding: 5px; margin-right: 10px;" type="button" value="Add zone"/> <input type="button" value="Cancel"/>		

## 验证



检查 **Firewall** 部分中的配置：

Networking > Firewall

<b>Firewall</b> <input checked="" type="checkbox"/>	Enabled	Incoming requests are blocked by default. Outgoing requests are not blocked.	<input type="button" value="Add new zone"/>																																				
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Home Zone</th> <th>Interface</th> <th>Allowed addresses</th> <th>Action</th> </tr> </thead> <tbody> <tr> <td>Home Zone</td> <td>Interface enp0s31f6</td> <td>Entire subnet</td> <td><input type="button" value="Add services"/></td> </tr> <tr> <td colspan="4"> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Service</th> <th>TCP</th> <th>UDP</th> <th>Action</th> </tr> </thead> <tbody> <tr> <td>ssh</td> <td>22</td> <td></td> <td>⋮</td> </tr> <tr> <td>mdns</td> <td></td> <td>5353</td> <td>⋮</td> </tr> <tr> <td>samba-client</td> <td></td> <td>137,138</td> <td>⋮</td> </tr> <tr> <td>dhcpcv6-client</td> <td></td> <td>546</td> <td>⋮</td> </tr> <tr> <td>cockpit</td> <td>9090</td> <td></td> <td>⋮</td> </tr> </tbody> </table> </td> </tr> </tbody> </table>				Home Zone	Interface	Allowed addresses	Action	Home Zone	Interface enp0s31f6	Entire subnet	<input type="button" value="Add services"/>	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Service</th> <th>TCP</th> <th>UDP</th> <th>Action</th> </tr> </thead> <tbody> <tr> <td>ssh</td> <td>22</td> <td></td> <td>⋮</td> </tr> <tr> <td>mdns</td> <td></td> <td>5353</td> <td>⋮</td> </tr> <tr> <td>samba-client</td> <td></td> <td>137,138</td> <td>⋮</td> </tr> <tr> <td>dhcpcv6-client</td> <td></td> <td>546</td> <td>⋮</td> </tr> <tr> <td>cockpit</td> <td>9090</td> <td></td> <td>⋮</td> </tr> </tbody> </table>				Service	TCP	UDP	Action	ssh	22		⋮	mdns		5353	⋮	samba-client		137,138	⋮	dhcpcv6-client		546	⋮	cockpit	9090		⋮
Home Zone	Interface	Allowed addresses	Action																																				
Home Zone	Interface enp0s31f6	Entire subnet	<input type="button" value="Add services"/>																																				
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Service</th> <th>TCP</th> <th>UDP</th> <th>Action</th> </tr> </thead> <tbody> <tr> <td>ssh</td> <td>22</td> <td></td> <td>⋮</td> </tr> <tr> <td>mdns</td> <td></td> <td>5353</td> <td>⋮</td> </tr> <tr> <td>samba-client</td> <td></td> <td>137,138</td> <td>⋮</td> </tr> <tr> <td>dhcpcv6-client</td> <td></td> <td>546</td> <td>⋮</td> </tr> <tr> <td>cockpit</td> <td>9090</td> <td></td> <td>⋮</td> </tr> </tbody> </table>				Service	TCP	UDP	Action	ssh	22		⋮	mdns		5353	⋮	samba-client		137,138	⋮	dhcpcv6-client		546	⋮	cockpit	9090		⋮												
Service	TCP	UDP	Action																																				
ssh	22		⋮																																				
mdns		5353	⋮																																				
samba-client		137,138	⋮																																				
dhcpcv6-client		546	⋮																																				
cockpit	9090		⋮																																				

### 41.7.10. 使用 Web 控制台禁用区域

您可以使用 Web 控制台在防火墙配置中禁用防火墙区域。

## 前提条件

- 您已安装了 RHEL 8 web 控制台。
- 您已启用了 cockpit 服务。
- 您的用户帐户被允许登录到 web 控制台。

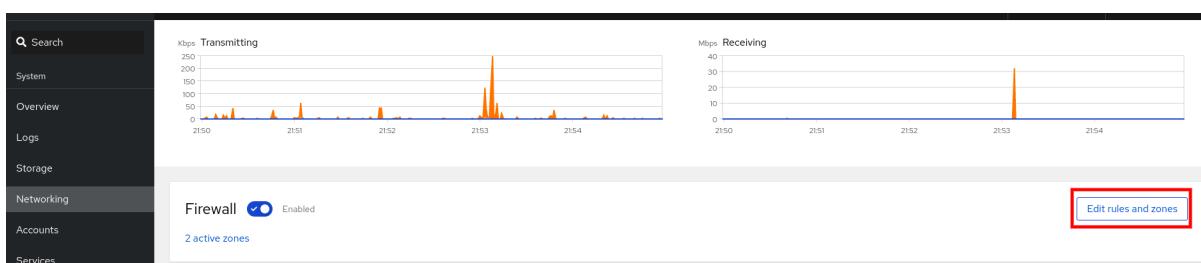
具体步骤请参阅[安装并启用 Web 控制台](#)。

## 流程

1. 登录到 RHEL 8 web 控制台。

详情请参阅[登录到 web 控制台](#)。

2. 点 Networking。
3. 点编辑规则和区域按钮。



如果没有看到 Edit rules and zones 按钮，使用管理员权限登录到 web 控制台。

4.

点您要删除的区的 Options 图标。

Service	TCP	UDP
ssh	22	
mdns		5353
samba-client		137, 138
dhcpv6-client		546
cockpit	9090	

5.

点击 **Delete**。

区域现在被禁用，接口不包括在区域中配置的打开的服务和端口。

#### 41.7.11. 使用区目标设定传入流量的默认行为

对于每个区，您可以设置一种处理尚未进一步指定的传入流量的默认行为。此行为是通过设置区的目标来定义的。有四个选项：

- **ACCEPT**：接受所有传入的数据包，除了特定规则禁止的。
- **REJECT**：拒绝所有传入的数据包，除了特定规则允许的。当 `firewalld` 拒绝数据包时，会告知源机器有关拒绝的信息。
- **DROP**：丢弃所有传入的数据包，除了特定规则允许的。当 `firewalld` 丢弃数据包时，不会告知源机器有关丢弃数据包的信息。
- **default**：与 **REJECT** 的行为类似，但在某些情况下具有特殊含义。

#### 先决条件

- **firewalld** 服务在运行。

## 流程

为区设置目标：

1. 列出特定区的信息以查看默认目标：

```
firewall-cmd --zone=zone-name --list-all
```

2. 在区中设置一个新目标：

```
firewall-cmd --permanent --zone=zone-name --set-target=<default|ACCEPT|REJECT|DROP>
```

## 其他资源

- 您系统上的 **firewall-cmd (1)** 手册页

### 41.7.12. 配置动态更新，以允许 IP 集合

您可以进行接近实时更新，来灵活地允许 IP 集中特定的 IP 地址或范围，即使在无法预计的情况下也是如此。这些更新可由各种事件触发，如安全威胁检测或网络行为的更改。通常，此类解决方案利用自动化来减少手工工作，并通过快速响应情况来提高安全性。

## 先决条件

- **firewalld** 服务在运行。

## 流程

1. 创建一个有有意义名称的 IP 集：

```
firewall-cmd --permanent --new-ipset=allowlist --type=hash:ip
```

名为 **allowlist** 的新 IP 集包含您希望防火墙允许的 IP 地址。

2.

向 IP 集添加一个动态更新：

```
firewall-cmd --permanent --ipset=allowlist --add-entry=198.51.100.10
```

此配置使用新添加的、防火墙允许传输网络流量的 IP 地址更新 **allowlist** IP 集。

3.

创建一个引用之前创建的 IP 集的防火墙规则：

```
firewall-cmd --permanent --zone=public --add-source=ipset:allowlist
```

没有此规则，IP 集不会对网络流量有任何影响。以默认防火墙策略为准。

4.

重新载入防火墙配置，以应用更改：

```
firewall-cmd --reload
```

## 验证

1.

列出所有 IP 集：

```
firewall-cmd --get-ipsets
allowlist
```

2.

列出活跃的规则：

```
firewall-cmd --list-all
public (active)
target: default
icmp-block-inversion: no
interfaces: enp0s1
sources: ipset:allowlist
services: cockpit dhcpcv6-client ssh
ports:
protocols:
...
```

命令行输出的 **sources** 部分提供了哪些流量的来源（主机名、接口、IP 集、子网等）被允许或拒绝访问特定防火墙区域的信息。在这种情况下，允许 **allowlist** IP 集中包含的 IP 地址通过 **public** 区域的防火墙传输流量。

3.

探索 IP 集的内容：

```
cat /etc/firewalld/ipsets/allowlist.xml
<?xml version="1.0" encoding="utf-8"?>
<ipset type="hash:ip">
 <entry>198.51.100.10</entry>
</ipset>
```

## 后续步骤

- 

使用脚本或安全工具来获取您的威胁情报源，并以自动化方式更新 **allowlist**。

## 其他资源

- 

[firewall-cmd \(1\)](#) 手册页

## 41.8. 使用 FIREWALLD 控制网络流量

**firewalld** 软件包安装了大量预定义的服务文件，您可以添加更多或自定义它们。然后，您可以使用这些服务定义为服务打开或关闭端口，而无需了解协议及它们使用的端口号。

### 41.8.1. 使用 CLI 控制预定义服务的流量

控制流量的最简单的方法是在 **firewalld** 中添加预定义的服务。这会打开所有必需的端口并根据 **服务定义文件** 修改其他设置。

#### 先决条件

- 

**firewalld** 服务在运行。

#### 流程

1.

检查 `firewall` 中的服务是否没有被允许：

```
firewall-cmd --list-services
ssh dhcpcv6-client
```

命令列出默认区域中启用的服务。

2.

列出 `firewall` 中所有预定义的服务：

```
firewall-cmd --get-services
RH-Satellite-6 amanda-client amanda-k5-client bacula bacula-client bitcoin bitcoin-rpc
bitcoin-testnet bitcoin-testnet-rpc ceph ceph-mon cfengine condor-collector ctdb dhcp
dhcpcv6 dhcpcv6-client dns docker-registry ...
```

命令显示默认区域的可用服务的列表。

3.

将服务添加到 `firewall` 允许的服务的列表中：

```
firewall-cmd --add-service=<service_name>
```

命令将指定的服务添加到默认区域中。

4.

使新设置具有持久性：

```
firewall-cmd --runtime-to-permanent
```

命令将这些运行时更改应用到防火墙的永久配置中。默认情况下，它将这些更改应用到默认区域的配置中。

## 验证

1.

列出所有永久防火墙规则：

```
firewall-cmd --list-all --permanent
public
```

```

target: default
icmp-block-inversion: no
interfaces:
sources:
services: cockpit dhcpcv6-client ssh
ports:
protocols:
forward: no
masquerade: no
forward-ports:
source-ports:
icmp-blocks:
rich rules:

```

命令显示带有默认防火墙区域(**public**)的永久防火墙规则的完整配置。

2. 检查 **firewalld** 服务的永久配置的有效性。

```

firewall-cmd --check-config
success

```

如果永久配置无效，命令返回一个带有更多详情的错误：

```

firewall-cmd --check-config
Error: INVALID_PROTOCOL: 'public.xml': 'tcpx' not from {'tcp'|'udp'|'sctp'|'dccp'}

```

您还可以手动检查永久配置文件，以验证设置。主配置文件为 **/etc/firewalld/firewalld.conf**。特定于区域的配置文件位于 **/etc/firewalld/zones/** 目录中，策略位于 **/etc/firewalld/policies/** 目录中。

#### 41.8.2. 使用 Web 控制台在防火墙上启用服务

默认情况下，服务添加到默认防火墙区。如果在更多网络接口中使用更多防火墙区，您必须首先选择一个区域，然后添加带有端口的服务。

RHEL 8 web 控制台显示预定义的 **firewalld** 服务，您可以将其添加到活跃的防火墙区。



## 重要

**RHEL 8 web 控制台配置 firewalld 服务。**

Web 控制台不允许没有在 web 控制台中列出的通用 firewalld 规则。

## 先决条件

- 您已安装了 RHEL 8 web 控制台。
- 您已启用了 cockpit 服务。
- 您的用户帐户被允许登录到 web 控制台。

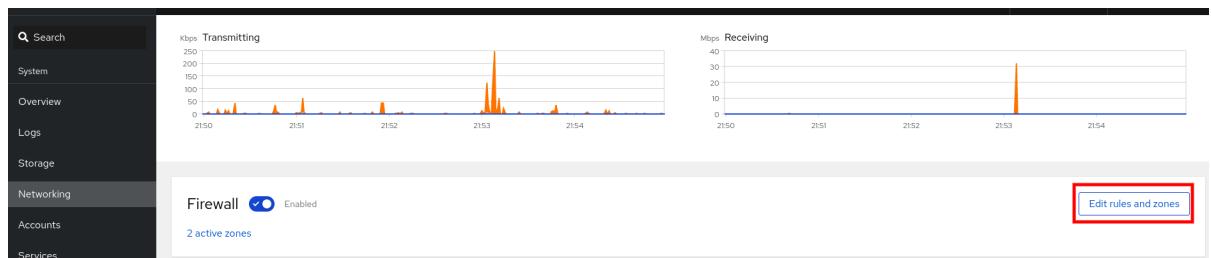
具体步骤请参阅[安装并启用 Web 控制台](#)。

## 流程

1. 登录到 RHEL 8 web 控制台。

详情请参阅[登录到 web 控制台](#)。

2. 点 Networking。
3. 点编辑规则和区域按钮。



如果没有看到 **Edit rules and zones** 按钮，使用管理员权限登录到 web 控制台。

4.

在 **Firewall** 部分，选择要添加该服务的区，然后点击 **Add Services**。

The screenshot shows the 'Networking > Firewall' section. At the top, there's a status bar with 'Firewall' (Enabled), 'Incoming requests are blocked by default. Outgoing requests are not blocked.', and a 'Add new zone' button. Below this is a table titled 'Home Zone' showing various services and their ports. The 'Add services' button in the top right corner of the table area is highlighted with a red box.

Service	TCP	UDP
ssh	22	
mdns		5353
samba-client		137,138
dhcpv6-client		546
cockpit	9090	

5.

在 **Add Services** 对话框中，找到您要在防火墙中启用的服务。

6.

根据您的场景启用服务：

The screenshot shows the 'Add services to home zone' dialog. It has two radio button options: 'Services' (selected) and 'Custom ports'. A 'Filter services' input field contains 'freeIPA'. Below it is a list of services with checkboxes:
 

- freeipa-4  
TCP: 80, 443, 88, 464, 389, 636 UDP: 88, 464
- freeipa-ldap  
TCP: 80, 443, 88, 464, 389 UDP: 88, 464, 123
- freeipa-ldaps  
TCP: 80, 443, 88, 464, 636 UDP: 88, 464, 123
- freeipa-replication

 At the bottom are 'Add services' and 'Cancel' buttons.

7.

点 **Add Services**。

此时，RHEL 8 web 控制台在区域的服务列表中显示该服务。

#### 41.8.3. 使用 Web 控制台配置自定义端口

您可以通过 RHEL web 控制台为服务添加自定义端口。

##### 先决条件

- 您已安装了 RHEL 8 web 控制台。
- 您已启用了 cockpit 服务。
- 您的用户帐户被允许登录到 web 控制台。

具体步骤请参阅[安装并启用 Web 控制台](#)。

- firewalld 服务在运行。

##### 流程

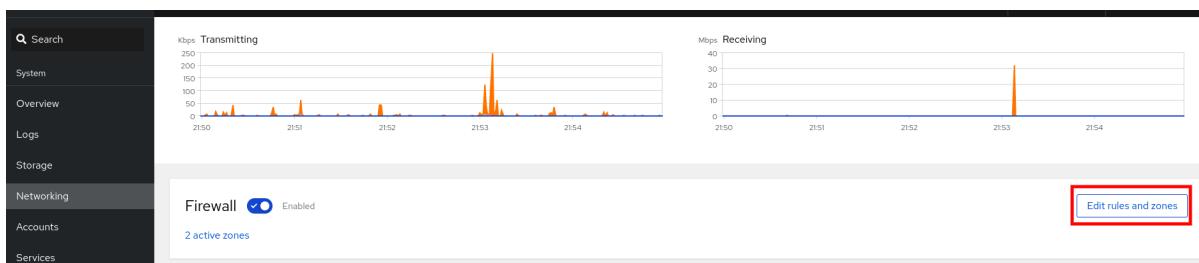
1. 登录到 RHEL 8 web 控制台。

详情请参阅[登录到 web 控制台](#)。

2. 点 **Networking**。

3.

点编辑规则和区域按钮。



如果没有看到 **Edit rules and zones** 按钮，使用管理员权限登录到 web 控制台。

4.

在 **Firewall** 部分，选择要配置自定义端口的区域，并点 **Add Services**。

Service	TCP	UDP
ssh	22	
mdns		5353
samba-client		137,138
dhcpv6-client		546
cockpit	9090	

5.

在 **Add services** 对话框中，点 **Custom Ports** 单选按钮。

6.

在 **TCP** 和 **UDP** 字段中，根据示例添加端口。您可以使用以下格式添加端口：

- 端口号，如 22
- 端口号范围，如 5900-5910

- 别名，比如 nfs, rsync

**注意**

您可以在每个字段中添加多个值。值必须用逗号分开，且没有空格，例如：  
8080、8081、http

7.

在 TCP 文件、UDP 文件或两者中添加端口号后，在 Name 字段中验证服务名称。

**Name** 字段显示保留此端口的服务名称。如果您确定这个端口可用，且不需要在该端口上通信，则可以重写名称。

8.

在 Name 字段中，为服务添加一个名称，包括定义的端口。

9.

点 **添加端口** 按钮。

### Add ports to home zone

x

Services  Custom ports

TCP

Example: 22,ssh,8080,5900-5910

Comma-separated ports, ranges, and services are accepted

UDP

Example: 88,2019,nfs,rsync

Comma-separated ports, ranges, and services are accepted

ID

If left empty, ID will be generated based on associated port services and port numbers

Description

**⚠ Adding custom ports will reload firewalld. A reload will result in the loss of any runtime-only configuration!**

**Add ports**

**Cancel**

要验证设置，请进入防火墙页面，并在区域的服务列表中找到该服务。

Service	TCP	UDP
ssh	22	
mdns		5353
samba-client		137,138
dhcpcv6-client		546
cockpit	9090	

## 41.9. 在区域间过滤转发的流量

**firewalld** 使您能够控制不同 **firewalld** 区域之间的网络数据流。通过定义规则和策略，您可以管理当流量在这些区域之间移动时，它们是如何被允许或阻止的。

策略对象功能在 **firewalld** 中提供转发和输出过滤。您可以使用 **firewalld** 过滤不同区域之间的流量，以允许访问本地托管的虚拟机，来连接主机。

### 41.9.1. 策略对象和区域之间的关系

策略对象允许用户将 **firewalld** 的原语（如服务、端口和富规则）附加到策略。您可以将策略对象应用到以有状态和单向的方式在区域间传输的流量上。

```
firewall-cmd --permanent --new-policy myOutputPolicy
firewall-cmd --permanent --policy myOutputPolicy --add-ingress-zone HOST
firewall-cmd --permanent --policy myOutputPolicy --add-egress-zone ANY
```

HOST 和 ANY 是 ingress 和 egress 区域列表中使用的符号区域。

- 

HOST 符号区域对于来自运行 **firewalld** 的主机的流量，或具有到运行 **firewalld** 的主机的流量允许策略。

- ANY 符号区对所有当前和将来的区域应用策略。ANY 符号区域充当所有区域的通配符。

#### 41.9.2. 使用优先级对策略进行排序

多个策略可以应用到同一组流量，因此应使用优先级为可能应用的策略创建优先级顺序。

要设置优先级来对策略进行排序：

```
firewall-cmd --permanent --policy mypolicy --set-priority -500
```

在上例中，-500 是较低的优先级值，但具有较高的优先级。因此，-500 将在 -100 之前执行。

较低的数字优先级值有较高的优先级，并被首先应用。

#### 41.9.3. 使用策略对象过滤本地托管的容器和物理连接到主机的网络之间的流量

策略对象功能允许用户过滤 Podman 和 firewalld 区域之间的流量。



##### 注意

红帽建议默认阻止所有流量，并打开 Podman 工具所需的可选择的服务。

##### 流程

1.

创建一个新的防火墙策略：

```
firewall-cmd --permanent --new-policy podmanToAny
```

2.

阻止从 Podman 到其它区域的所有流量，并只允许 Podman 上必要的服务：

```
firewall-cmd --permanent --policy podmanToAny --set-target REJECT
firewall-cmd --permanent --policy podmanToAny --add-service dhcp
```

```
firewall-cmd --permanent --policy podmanToAny --add-service dns
firewall-cmd --permanent --policy podmanToAny --add-service https
```

3.

创建一个新的 Podman 区域：

```
firewall-cmd --permanent --new-zone=podman
```

4.

为策略定义 ingress 区域：

```
firewall-cmd --permanent --policy podmanToHost --add-ingress-zone podman
```

5.

为所有其他区域定义 egress 区域：

```
firewall-cmd --permanent --policy podmanToHost --add-egress-zone ANY
```

将 egress 区域设置为 ANY 意味着您从 Podman 过滤到其他区域。如果要过滤到主机，那么将 egress 区域设置为 HOST。

6.

重启 firewalld 服务：

```
systemctl restart firewalld
```

验证

●

验证到其他区域的 Podman 防火墙策略：

```
firewall-cmd --info-policy podmanToAny
podmanToAny (active)
...
target: REJECT
ingress-zones: podman
egress-zones: ANY
services: dhcp dns https
...
```

#### 41.9.4. 设置策略对象的默认目标

您可以为策略指定 --set-target 选项。可用的目标如下：

- ACCEPT - 接受数据包
- DROP - 丢弃不需要的数据包
- REJECT - 拒绝不需要的数据包，并带有 ICMP 回复
- CONTINUE (默认) - 数据包将遵循以下策略和区域中的规则。

```
firewall-cmd --permanent --policy mypolicy --set-target CONTINUE
```

## 验证

- 验证有关策略的信息

```
firewall-cmd --info-policy mypolicy
```

## 41.10. 使用 FIREWALLD 配置 NAT

使用 firewalld，您可以配置以下网络地址转换(NAT)类型：

- 伪装
- 目标 NAT (DNAT)
- 重定向

### 41.10.1. 网络地址转换类型

这些是不同的网络地址转换 (NAT) 类型：

伪装

使用以上 NAT 类型之一更改数据包的源 IP 地址。例如，互联网服务提供商(ISP)不会路由私有 IP 范围，如 10.0.0.0/8。如果您在网络中使用私有 IP 范围，用户应该能够访问互联网上的服务器，请将来自这些范围的数据包的源 IP 地址映射为公共 IP 地址。

伪装自动使用传出接口的 IP 地址。因此，如果传出接口使用了动态 IP 地址，则使用伪装。

## 目标 NAT (DNAT)

使用此 NAT 类型重写传入数据包的目标地址和端口。例如，如果您的 Web 服务器使用来自私有 IP 范围的 IP 地址，因此无法直接从互联网访问，您可以在路由器上设置 DNAT 规则，来将传入的流量重定向到此服务器。

## 重定向

这个类型是 DNAT 的一种特殊情况，其将数据包重定向到本地计算机上的不同端口。例如，如果服务运行在与其标准端口不同的端口上，您可以将传入的流量从标准端口重定向到此特定端口。

### 41.10.2. 配置 IP 地址伪装

您可以在系统上启用 IP 伪装。在访问互联网时，IP 伪装会隐藏网关后面的单个机器。

## 流程

1.

要检查是否启用了 IP 伪装（例如，对于 external 区），以 root 用户身份输入以下命令：

```
firewall-cmd --zone=external --query-masquerade
```

如果已启用，命令将会打印 yes，且退出状态为 0。否则，将打印 no，且退出状态为 1。如果省略了 zone，则将使用默认区。

2.

要启用 IP 伪装，请以 root 用户身份输入以下命令：

```
firewall-cmd --zone=external --add-masquerade
```

3.

要使此设置持久，请将 --permanent 选项传给命令。

4.

要禁用 IP 伪装，请以 root 身份输入以下命令：

```
firewall-cmd --zone=external --remove-masquerade
```

要使此设置永久生效，请将 **--permanent** 选项传给命令。

#### 41.10.3. 使用 DNAT 转发传入的 HTTP 流量

您可以使用目标网络地址转换(DNAT)将传入的流量从一个目标地址和端口定向到另一个目标地址和端口。通常，这对于将来自外部网络接口的传入请求重定向到特定的内部服务器或服务非常有用。

##### 先决条件

- **firewalld** 服务在运行。

##### 流程

1. 转发传入的 HTTP 流量：

```
firewall-cmd --zone=public --add-forward-port=port=80:proto=tcp:toaddr=198.51.100.10:toport=8080 --permanent
```

之前的命令使用以下设置定义 DNAT 规则：

- **--zone=public** - 您为其配置 DNAT 规则的防火墙区域。您可以将其调整到您需要的任何区域。
- **--add-forward-port** - 表示您要添加一个端口转发规则的选项。
- **port=80** - 外部目标端口。
- **proto=tcp** - 表示您转发 TCP 流量的协议。

- **toaddr=198.51.100.10** - 目标 IP 地址。
- **toport=8080** - 内部服务器的目标端口。
- **--permanent** - 使 DNAT 规则在重启后持久的选项。

## 2. 重新载入防火墙配置，以应用更改：

```
firewall-cmd --reload
```

验证

● 验证您使用的防火墙区域的 DNAT 规则：

```
firewall-cmd --list-forward-ports --zone=public
port=80:proto=tcp:toport=8080:toaddr=198.51.100.10
```

或者，查看相应的 XML 配置文件：

```
cat /etc/firewalld/zones/public.xml
<?xml version="1.0" encoding="utf-8"?>
<zone>
 <short>Public</short>
 <description>For use in public areas. You do not trust the other computers on
 networks to not harm your computer. Only selected incoming connections are
 accepted.</description>
 <service name="ssh"/>
 <service name="dhcpcv6-client"/>
 <service name="cockpit"/>
 <forward-port port="80" protocol="tcp" to-port="8080" to-addr="198.51.100.10"/>
 </forward>
</zone>
```

其他资源

- [在运行时配置内核参数](#)

- [firewall-cmd \(1\) 手册页](#)

#### 41.10.4. 重定向来自非标准端口的流量，以使 Web 服务在标准端口上可访问

您可以使用重定向机制使内部运行在非标准端口上的 Web 服务可访问，而无需用户在 URL 中指定端口。因此，URL 更为简单，提供更好的浏览体验，而非标准端口仍然在内部使用或用于特定的要求。

##### 前提条件

- [firewalld 服务在运行。](#)

##### 流程

1. [创建 NAT 重定向规则：](#)

```
firewall-cmd --zone=public --add-forward-port=port=<standard_port>:proto=tcp:toport=<non_standard_port> --permanent
```

之前的命令使用以下设置定义 NAT 重定向规则：

- **--zone=public** - 您为其配置规则的防火墙区域。您可以将它调整为您需要的任何区域。
- **--add-forward-port=port=<non\_standard\_port>** - 表示您使用最初接收传入流量的源端口添加一个端口转发（重定向）规则。
- **proto=tcp** - 表示您重定向 TCP 流量的协议。
- **toport=<standard\_port>** - 在源端口上收到传入流量后，应将其重定向到的目标端口。
- **--permanent** - 使规则在重启后持久的选项。

2.

重新载入防火墙配置，以应用更改：

```
firewall-cmd --reload
```

验证

- 验证您使用的防火墙区域的重定向规则：

```
firewall-cmd --list-forward-ports
port=8080:proto=tcp:toport=80:toaddr=
```

或者，查看相应的 XML 配置文件：

```
cat /etc/firewalld/zones/public.xml
<?xml version="1.0" encoding="utf-8"?>
<zone>
 <short>Public</short>
 <description>For use in public areas. You do not trust the other computers on
networks to not harm your computer. Only selected incoming connections are
accepted.</description>
 <service name="ssh"/>
 <service name="dhcpcv6-client"/>
 <service name="cockpit"/>
 <forward-port port="8080" protocol="tcp" to-port="80"/>
 <forward/>
</zone>
```

其他资源

- [在运行时配置内核参数](#)
- [firewall-cmd \(1\) 手册页](#)

#### 41.11. 丰富规则的优先级

富规则提供了一种更高级且更灵活的方法来定义防火墙规则。富规则特别有用，其中服务、端口等服务不足以表达复杂的防火墙规则。

富规则背后的概念：

## 粒度和灵活性

您可以根据更为具体的标准为网络流量定义详细条件。

## 规则结构

富规则由家族(IPv4 或 IPv6)组成，后跟条件和操作。

```
rule family="ipv4|ipv6" [conditions] [actions]
```

### conditions

它们允许富规则仅在符合特定条件时才适用。

### 操作

您可以定义与条件匹配的网络流量发生的情况。

### 组合多个条件

您可以创建更为具体的和复杂的过滤。

### 分层控制和可重复利用

您可以将丰富的规则与其他防火墙机制（如区域或服务）相结合。

默认情况下，富规则是根据其规则操作进行组织的。例如，`deny` 规则优先于 `allow` 规则。富规则中的 `priority` 参数可让管理员对富规则及其执行顺序进行精细的控制。在使用 `priority` 参数时，规则首先按其优先级值升序排序。当许多规则有相同的 `priority` 时，其顺序是由规则操作决定的，如果操作也相同，则顺序可能未定义。

#### 41.11.1. priority 参数如何将规则组织为不同的链

您可以将富规则中的 `priority` 参数设置为 -32768 和 32767 之间的任何数，较小的数字值有较高的优先级。

`firewalld` 服务根据优先级值将规则组织到不同的链中：

- 优先级低于 0：规则被重定向到带有 `_pre` 后缀的链中。
- 优先级高于 0：规则被重定向到带有 `_post` 后缀的链中。
- 优先级等于 0：根据操作，规则会被重定向到带有 `_log`、`_deny` 或 `_allow` 操作的链中。

在这些子链中，`firewalld` 根据其优先级值对规则进行排序。

## 其他资源

- `firewalld.richlanguage(5)`

### 41.11.2. 设置丰富的规则的优先级

以下是如何创建一条富规则的示例，该规则使用 `priority` 参数来记录其他规则不允许或拒绝的所有流量。您可以使用此规则标记意非预期的流量。

## 流程

- 添加一个带有非常低优先级的丰富规则来记录未由其他规则匹配的所有流量：

```
firewall-cmd --add-rich-rule='rule priority=32767 log prefix="UNEXPECTED: " limit value="5/m"'
```

这个命令还将日志条目数量限制为每分钟 5 条。

## 验证

- 显示命令在上一步中创建的 `nftables` 规则：

```
nft list chain inet firewalld filter_IN_public_post
table inet firewalld {
 chain filter_IN_public_post {
```

```
 log prefix "UNEXPECTED: " limit rate 5/minute
}
```

## 其他资源

- [`firewalld.richlanguage\(5\)](#)

### 41.12. 启用 FIREWALLD 区域中不同接口或源之间的流量转发

区内转发是 `firewalld` 的一种功能，它允许 `firewalld` 区域内接口或源之间的流量转发。

#### 41.12.1. 区内转发与默认目标设置为 ACCEPT 的区域之间的区别

启用区域内转发后，单个 `firewalld` 区域中的流量可以从一个接口或源流到另一个接口或源。区指定接  
口和源的信任级别。如果信任级别相同，流量会保持在同一区域内。



#### 注意

在 `firewalld` 的默认区域中启用区域内转发仅适用于添加到当前默认区域的接口和源。

`firewalld` 使用不同的区域管理传入和传出流量。每个区域都有自己的一组规则和行为。例如，`trusted` 区域默认允许所有转发的流量。

其他区域可以有不同的默认行为。在标准区域中，当区域的目标被设置为 `default` 时，转发的流量通常  
默认被丢弃。

要控制流量如何在区域内的不同接口或源之间转发，请确保您理解并相应地配置了区域的目标。

#### 41.12.2. 使用区内转发来在以太网和 Wi-Fi 网络间转发流量

您可以使用区内转发来在同一 `firewalld` 区内的接口和源之间转发流量。此功能带来以下好处：

- 有线设备和无线设备间的无缝连接（您可以在连接到 `enp1s0` 的以太网网络和连接到

wlp0s20 的 Wi-Fi 网络之间转发流量)

- 支持灵活的工作环境
- 可被网络中的多个设备或用户访问和使用的共享资源（如打印机、数据库、网络连接的存储等）
- 高效的内部网络（如顺畅通信、减少的延迟、资源可访问性等）

您可以为单个 `firewalld` 区域启用此功能。

## 流程

1.

在内核中启用数据包转发：

```
echo "net.ipv4.ip_forward=1" > /etc/sysctl.d/95-IPv4-forwarding.conf
sysctl -p /etc/sysctl.d/95-IPv4-forwarding.conf
```

2.

确保您要在其间启用区域内转发的接口只分配到 `internal` 区域：

```
firewall-cmd --get-active-zones
```

3.

如果接口当前被分配给了不是 `internal` 的区，请重新分配它：

```
firewall-cmd --zone=internal --change-interface=interface_name --permanent
```

4.

将 `enp1s0` 和 `wlp0s20` 接口添加到 `internal` 区：

```
firewall-cmd --zone=internal --add-interface=enp1s0 --add-interface=wlp0s20
```

5.

启用区域内部转发：

```
firewall-cmd --zone=internal --add-forward
```

## 验证

以下验证要求 nmap-ncat 软件包已安装在两个主机上。

1.

登录到与您启用了区域转发的主机的 `enp1s0` 接口位于同一网络的主机。

2.

使用 `ncat` 启动 `echo` 服务来测试连接：

```
ncat -e /usr/bin/cat -l 12345
```

3.

登录到与 `wlp0s20` 接口在同一网络的主机。

4.

连接到在与 `enp1s0` 在同一网络的主机上运行的 `echo` 服务器：

```
ncat <other_host> 12345
```

5.

输入一些内容并按 `Enter`。验证文本是否被发送回来。

## 其他资源



您系统上的 `firewalld.zones (5)` 手册页

### 41.13. 使用 RHEL 系统角色配置 FIREWALLD

RHEL 系统角色是 Ansible 自动化工具的一组内容。此内容与 Ansible 自动化工具一起提供一个一致的配置接口，来一次远程管理多个系统。

`rhel-system-roles` 软件包包含 `rhel-system-roles.firewall` RHEL 系统角色。此角色是为了自动化 `firewalld` 服务的配置而引入的。

使用 `firewall` RHEL 系统角色，您可以配置许多不同的 `firewalld` 参数，例如：

- 区域
- 应允许哪些数据包的服务
- 授予、拒绝或丢弃对端口的流量访问
- 区域的端口或端口范围的转发

#### 41.13.1. 使用 firewall RHEL 系统角色重置 firewalld 设置

随着时间的推移，更新您的防火墙配置可能会积累到一定程度，从而可能导致意外的安全风险。使用 **firewall RHEL 系统角色**，您可以以自动的方式将 **firewalld** 设置重置为其默认状态。这样，您可以有效地删除无意的或不安全的防火墙规则，并简化其管理。

#### 先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 **playbook** 的用户登录到控制节点。
- 您用于连接到受管节点的帐户对它们具有 **sudo** 权限。

#### 流程

1. 创建一个包含以下内容的 **playbook** 文件，如 `~/playbook.yml`：

```

- name: Reset firewalld example
 hosts: managed-node-01.example.com
 tasks:
 - name: Reset firewalld
 ansible.builtin.include_role:
 name: redhat.rhel_system_roles.firewall
```

```
vars:
 firewall:
 - previous: replaced
```

示例 playbook 中指定的设置包括如下：

**previous: replaced**

删除所有现有的用户定义的设置，并将 firewalld 设置重置为默认值。如果将 **previous:replaced** 参数与其他设置相结合，则 **firewall** 角色会在应用新设置前删除所有现有设置。

有关 playbook 中使用的所有变量的详情，请查看控制节点上的 **/usr/share/ansible/roles/rhel-system-roles.firewall/README.md** 文件。

2.

验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不能防止错误的、但有效的配置。

3.

运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

•

在控制节点上运行这个命令，来远程检查受管节点上的所有防火墙配置是否已被重置为其默认值：

```
ansible managed-node-01.example.com -m ansible.builtin.command -a 'firewall-cmd
--list-all-zones'
```

其他资源

•

**/usr/share/ansible/roles/rhel-system-roles.firewall/README.md** 文件

- /usr/share/doc/rhel-system-roles/firewall/ 目录

#### 41.13.2. 使用 firewall RHEL 系统角色，将 firewalld 中的传入流量从一个本地端口转发到不同的本地端口

您可以使用 **firewall RHEL** 系统角色进行远程配置，将传入流量从一个本地端口转发到不同的本地端口。

例如，如果您有一个环境，其中同一机器上有多个服务共存且需要同样的默认端口，则可能会出现端口冲突。这些冲突可能会破坏服务并导致停机。使用 **firewall RHEL** 系统角色，您可以有效地将流量转发到替代端口，以确保您的服务可以同时运行，而无需修改其配置。

#### 先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 **playbook** 的用户登录到控制节点。
- 您用于连接到受管节点的帐户对它们具有 **sudo** 权限。

#### 流程

1. 创建一个包含以下内容的 **playbook** 文件，如 `~/playbook.yml`：

```

- name: Configure firewalld
 hosts: managed-node-01.example.com
 tasks:
 - name: Forward incoming traffic on port 8080 to 443
 ansible.builtin.include_role:
 name: redhat.rhel_system_roles.firewall
 vars:
 firewall:
 - forward_port: 8080/tcp;443;
 state: enabled
 runtime: true
 permanent: true
```

示例 playbook 中指定的设置包括如下：

**forward\_port: 8080/tcp;443**

使用 TCP 协议到达本地端口 8080 的流量被转发到端口 443。

**runtime: true**

启用运行时配置中的更改。默认值被设置为 true。

有关 playbook 中使用的所有变量的详情，请查看控制节点上的 /usr/share/ansible/roles/rhel-system-roles.firewall/README.md 文件。

2.

验证 playbook 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不能防止错误的、但有效的配置。

3.

运行 playbook：

```
$ ansible-playbook ~/playbook.yml
```

验证

•

在控制节点上，运行以下命令远程检查受管节点上的转发端口：

```
ansible managed-node-01.example.com -m ansible.builtin.command -a 'firewall-cmd --list-forward-ports'
managed-node-01.example.com | CHANGED | rc=0 >>
port=8080:proto=tcp:toport=443:toaddr=
```

其他资源

•

/usr/share/ansible/roles/rhel-system-roles.firewall/README.md 文件

- /usr/share/doc/rhel-system-roles/firewall/ 目录

#### 41.13.3. 使用 firewall RHEL 系统角色配置 firewalld DMZ 区域

作为系统管理员，您可以使用 **firewall RHEL** 系统角色在 `enp1s0` 接口上配置 `dmz` 区域，以允许到区的 `HTTPS` 流量。这样，您可以让外部用户访问您的 `web` 服务器。

##### 先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 `playbook` 的用户登录到控制节点。
- 您用于连接到受管节点的帐户对它们具有 `sudo` 权限。

##### 流程

1.

创建一个包含以下内容的 `playbook` 文件，如 `~/playbook.yml`：

```

- name: Configure firewalld
 hosts: managed-node-01.example.com
 tasks:
 - name: Creating a DMZ with access to HTTPS port and masquerading for hosts in
 DMZ
 ansible.builtin.include_role:
 name: redhat.rhel_system_roles.firewall
 vars:
 firewall:
 - zone: dmz
 interface: enp1s0
 service: https
 state: enabled
 runtime: true
 permanent: true
```

有关 `playbook` 中使用的所有变量的详情，请查看控制节点上的  
`/usr/share/ansible/roles/rhel-system-roles.firewall/README.md` 文件。

2.

验证 **playbook** 语法：

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

请注意，这个命令只验证语法，不能防止错误的、但有效的配置。

3.

运行 **playbook**：

```
$ ansible-playbook ~/playbook.yml
```

验证

●

在控制节点上，运行以下命令远程检查受管节点上 **dmz** 区域的信息：

```
ansible managed-node-01.example.com -m ansible.builtin.command -a 'firewall-cmd --zone=dmz --list-all'
managed-node-01.example.com | CHANGED | rc=0 >>
dmz (active)
 target: default
 icmp-block-inversion: no
 interfaces: enp1s0
 sources:
 services: https ssh
 ports:
 protocols:
 forward: no
 masquerade: no
 forward-ports:
 source-ports:
 icmp-blocks:
```

其他资源

●

[/usr/share/ansible/roles/rhel-system-roles.firewall/README.md](#) 文件

●

[/usr/share/doc/rhel-system-roles/firewall/](#) 目录

## 第 42 章 NFTABLES 入门

如果您的场景不在 `firewalld` 涵盖的典型数据包过滤情况下，或者您希望完全控制规则，您可以使用 `nftables` 框架。

`nftables` 框架对数据包进行分类，它是 `iptables`、`ip6tables`、`arptables`、`eptables` 和 `ipset` 工具的后续者。与之前的数据包过滤工具相比，它在方便、特性和性能方面提供了大量改进，最重要的是：

- 内置查找表而不是线性处理
- IPv4 和 IPv6 协议的单一框架
- 通过事务更新内核规则集，而不是获取、更新和存储整个规则集
- 支持在规则集(`nftrace`)和监控追踪事件 (`nft`) 中调试和追踪
- 更加一致和压缩的语法，没有特定协议的扩展
- 用于第三方应用程序的 Netlink API

`nftables` 框架使用表来存储链。链包含执行动作的独立规则。`nft` 工具替换了之前数据包过滤框架中的所有工具。您可以使用 `libnftables` 库通过 `libnftnl` 库与 `nftables` Netlink API 进行低级交互。

要显示规则集变化的影响，请使用 `nft list ruleset` 命令。要清除内核规则集，请使用 `nft flush ruleset` 命令。请注意，这也可能会影响 `iptables-nft` 命令安装的规则集，因为它使用了相同的内核基础架构。

### 42.1. 创建和管理 NFTABLES 表、链和规则

您可以显示 `nftables` 规则集并管理它们。

### 42.1.1. nftables 表的基础知识

nftables 中的表是一个包含链、规则、集合和其他对象集合的名字空间。

每个表都必须分配一个地址系列。地址系列定义此表处理的数据包类型。在创建表时，您可以设置以下地址系列之一：

- **ip**：仅匹配 IPv4 数据包。如果没有指定地址系列，这是默认设置。
- **ip6**：仅匹配 IPv6 数据包。
- **inet**：匹配 IPv4 和 IPv6 数据包。
- **arp**：匹配 IPv4 地址解析协议(ARP)数据包。
- **bridge**：匹配通过网桥设备的数据包。
- **netdev**：匹配来自入口的数据包。

如果要添加表，所使用的格式取决于您的防火墙脚本：

- 在原生语法的脚本中，使用：

```
table <table_address_family> <table_name> {
}
```

- 在 shell 脚本中，使用：

```
nft add table <table_address_family> <table_name>
```

### 42.1.2. nftables 链的基础知识

表由链组成，链又是规则的容器。存在以下两种规则类型：

- **Base chain**：您可以使用基本链作为来自网络堆栈的数据包的入口点。
- **Regular chain**：您可以使用常规链作为 jump 目标，来更好地组织规则。

如果要向表中添加基本链，所使用的格式取决于您的防火墙脚本：

- 在原生语法的脚本中，使用：

```
table <table_address_family> <table_name> {
 chain <chain_name> {
 type <type> hook <hook> priority <priority>
 policy <policy> ;
 }
}
```

- 在 shell 脚本中，使用：

```
nft add chain <table_address_family> <table_name> <chain_name> { type <type> hook
<hook> priority <priority> \; policy <policy> \; }
```

为了避免 shell 将分号解释为命令的结尾，请将 \ 转义字符放在分号前面。

这两个示例都创建 基本链。要创建 常规链，请不要在大括号中设置任何参数。

## 链类型

以下是链类型以及您可以使用的地址系列和钩子的概述：

类型	地址系列	Hook	描述
filter	all	all	标准链类型

类型	地址系列	Hook	描述
nat	ip,ip6/inet	prerouting、input 、output、postrout ing	这个类型的链根据连接跟踪条目执行原生地址 转换。只有第一个数据包会遍历此链类型。
route	ip,ip6	output	如果 IP 头的相关部分已更改，则接受的遍历此 链类型的数据包会导致新的路由查找。

## 链优先级

**priority** 参数指定数据包遍历具有相同 **hook** 值的链的顺序。您可以将此参数设为整数值，或使用标准优先级名称。

以下列表是标准优先级名称及其数字值的一个概述，以及您可以使用它们的哪个地址系列和钩子：

文本值	数字值	地址系列	钩子
raw	-300	ip,ip6/inet	all
mangle	-150	ip,ip6/inet	all
dstnat	-100	ip,ip6/inet	prerouting
	-300	bridge	prerouting
filter	0	ip,ip6/inet,arp,netdev	all
	-200	bridge	all
安全	50	ip,ip6/inet	all
srcnat	100	ip,ip6/inet	postrouting
	300	bridge	postrouting
out	100	bridge	output

## 链策略

如果此链中的规则没有指定任何操作，则链策略定义 **nftables** 是否应该接受或丢弃数据包。您可以在链中设置以下策略之一：

- accept (默认)
- drop

#### 42.1.3. nftables 规则的基础知识

规则定义对通过包含此规则的链的数据包执行的操作。如果规则还包含匹配表达式，则 **nftables** 仅在所有之前的表达式都应用时才执行操作。

如果要在链中添加一条规则，所使用的格式取决于您的防火墙脚本：

- 在原生语法的脚本中，使用：

```
table <table_address_family> <table_name> {
 chain <chain_name> {
 type <type> hook <hook> priority <priority> ; policy <policy> ;
 <rule>
 }
}
```

- 在 shell 脚本中，使用：

```
nft add rule <table_address_family> <table_name> <chain_name> <rule>
```

此 **shell** 命令在链的末尾附加新规则。如果要在链的开头添加一条规则，请使用 **nft insert** 命令而不是 **nft add**。

#### 42.1.4. 使用 nft 命令管理表、链和规则

要在命令行上或 **shell** 脚本中管理 **nftables** 防火墙，请使用 **nft** 工具。

**重要**

此流程中的命令不代表典型的工作流，且没有被优化。此流程只演示了如何使用 nft 命令来管理表、链和规则。

**流程**

1.

创建一个带有 inet 地址系列的名为 `nftables_svc` 的表，以便表可以处理 IPv4 和 IPv6 数据包：

```
nft add table inet nftables_svc
```

2.

将处理传入网络流量的、名为 INPUT 的基本链添加到 `inet nftables_svc` 表中：

```
nft add chain inet nftables_svc INPUT { type filter hook input priority filter; policy accept; }
```

为了避免 shell 将分号解释为命令的结尾，请使用 \ 字符转义分号。

3.

向 INPUT 链添加规则。例如，允许端口 22 和 443 上的传入 TCP 流量，并作为 INPUT 链的最后一条规则，拒绝其他传入的流量，并伴有互联网控制消息协议(ICMP)端口无法访问的消息：

```
nft add rule inet nftables_svc INPUT tcp dport 22 accept
nft add rule inet nftables_svc INPUT tcp dport 443 accept
nft add rule inet nftables_svc INPUT reject with icmpx type port-unreachable
```

如果您输入 `nft add rule` 命令，则 nft 会将按与运行命令相同的顺序将规则添加到链。

4.

显示包括句柄的当前规则集：

```
nft -a list table inet nftables_svc
table inet nftables_svc { # handle 13
 chain INPUT { # handle 1
 type filter hook input priority filter; policy accept;
 tcp dport 22 accept # handle 2
 tcp dport 443 accept # handle 3
 reject # handle 4
 }
}
```

5.

在句柄为 3 的现有规则前面插入一条规则。例如，要插入一个允许端口 636 上 TCP 流量的规则，请输入：

```
nft insert rule inet nftables_svc INPUT position 3 tcp dport 636 accept
```

6.

在句柄为 3 的现有规则后面附加一条规则。例如，要插入一个允许端口 80 上 TCP 流量的规则，请输入：

```
nft add rule inet nftables_svc INPUT position 3 tcp dport 80 accept
```

7.

再次显示带有 handle 的规则集。验证是否后添加的规则已添加到指定位置：

```
nft -a list table inet nftables_svc
table inet nftables_svc { # handle 13
 chain INPUT { # handle 1
 type filter hook input priority filter; policy accept;
 tcp dport 22 accept # handle 2
 tcp dport 636 accept # handle 5
 tcp dport 443 accept # handle 3
 tcp dport 80 accept # handle 6
 reject # handle 4
 }
}
```

8.

删除 handle 为 6 的规则：

```
nft delete rule inet nftables_svc INPUT handle 6
```

要删除规则，您必须指定 handle。

9.

显示规则集，并验证删除的规则是否不再存在：

```
nft -a list table inet nftables_svc
table inet nftables_svc { # handle 13
 chain INPUT { # handle 1
 type filter hook input priority filter; policy accept;
 tcp dport 22 accept # handle 2
 tcp dport 636 accept # handle 5
 tcp dport 443 accept # handle 3
 }
}
```

```
 }
 reject # handle 4
}
```

10.

从 **INPUT** 链中删除所有剩余的规则：

```
nft flush chain inet nftables_svc INPUT
```

11.

显示规则集，并验证 **INPUT** 链是否为空：

```
nft list table inet nftables_svc
table inet nftables_svc {
 chain INPUT {
 type filter hook input priority filter; policy accept
 }
}
```

12.

删除 **INPUT** 链：

```
nft delete chain inet nftables_svc INPUT
```

您还可以使用此命令删除仍然包含规则的链。

13.

显示规则集，并验证 **INPUT** 链是否已被删除：

```
nft list table inet nftables_svc
table inet nftables_svc {
}
```

14.

删除 **nftables\_svc** 表：

```
nft delete table inet nftables_svc
```

您还可以使用此命令删除仍然包含链的表。



### 注意

要删除整个规则集，请使用 `nft flush ruleset` 命令，而不是在单独的命令中手动删除所有规则、链和表。

## 其他资源

您系统上的 `nft (8)` 手册页

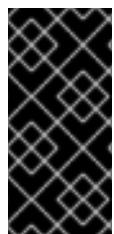
### 42.2. 从 IPTABLES 迁移到 NFTABLES

如果您的防火墙配置仍然使用 `iptables` 规则，则您可以将 `iptables` 规则迁移到 `nftables`。

#### 42.2.1. 使用 `firewalld`、`nftables` 或者 `iptables` 时

在 RHEL 8 中，您可以根据您的情况使用以下数据包过滤工具：

- `firewalld` : `firewalld` 工具简化了常见用例的防火墙配置。
- `nftables` : 使用 `nftables` 工具来设置复杂和性能关键的防火墙，如用于整个网络。
- `iptables` : Red Hat Enterprise Linux 上的 `iptables` 工具使用 `nf_tables` 内核 API 而不是传统的后端。`nf_tables` API 提供了向后兼容性，以便使用 `iptables` 命令的脚本仍可在 Red Hat Enterprise Linux 上工作。对于新的防火墙脚本，请使用 `nftables`。



### 重要

要防止不同的与防火墙相关的服务(`firewalld`、`nftables` 或 `iptables`)相互影响，请在 RHEL 主机上仅运行其中一个服务，并禁用其他服务。

#### 42.2.2. `nftables` 框架中的概念

与 `iptables` 框架相比，`nftables` 提供了更现代化、更高效且更灵活的替代方案。`nftables` 框架比 `iptables` 提供了高级功能和改进，从而简化规则管理和增强性能。这使得 `nftables` 成为复杂和高性能网

络环境的现代替代方案。

## 表和命名空间

在 **nftables** 中，表代表组织单元或命名空间，其将相关的防火墙链、集合、流表和其他对象分组在一起。在 **nftables** 中，表提供了一种更灵活的方法来构建防火墙规则和相关组件。在 **iptables** 中，表被更严格地定义，并具有特定用途。

## 表系列

**nftables** 中的每个表都与特定的系列(ip、ip6、inet、arp、bridge 或 netdev)关联。此关联决定了表可以处理哪些数据包。例如，ip 系列中的一个表只处理 IPv4 数据包。另一方面，inet 是表系列的一个特例。它提供了一个跨协议的统一方法，因为 IPv4 数据包和 IPv6 数据包它都可以处理。特殊表系列的另一种情况是 netdev，因为它用于直接应用到网络设备的规则，从而在设备级别上启用过滤。

## 基本链

**nftables** 中的基本链是数据包处理管道中高度可配置的入口点，允许用户指定以下内容：

- 链的类型，如 "filter"
- 数据包处理路径中的 hook 点，例如 "input", "output", "forward"
- 链的优先级

通过这种灵活性，可以精确控制规则在通过网络堆栈时何时及如何应用到数据包。链的一种特殊情况是 route 链，其用于根据数据包头，影响内核所做的路由决策。

## 用于规则处理的虚拟机

**nftables** 框架使用一个内部虚拟机来处理规则。此虚拟机执行与汇编语言操作类似的指令（将数据加载到寄存器，执行比较等）。这种机制可以实现高度灵活和有效的规则处理。

**nftables** 中的增强功能可以作为该虚拟机的新指令引入。这通常需要一个新的内核模块，以及对 libnftnl 库和 nft 命令行工具的更新。

或者，您可以通过将现有指令以一种创新的方式结合来引进新功能，而无需进行内核修

改。nftables 规则的语法反映了底层虚拟机的灵活性。例如，如果 TCP 目的地端口为 22，规则 `meta mark set tcp dport map { 22: 1, 80: 2 }` 将数据包的防火墙标记设置为 1，如果端口为 80，则设置为 2。这展示了如何简洁地表达复杂的逻辑。

## 复杂的过滤和字典映射

nftables 框架集成并扩展了 ipset 工具的功能，其在 iptables 中用于对 IP 地址、端口、其他数据类型、最重要的是其中的组合进行批量匹配。此集成使您更容易直接管理 nftables 中的大型和动态的数据集。接下来，nftables 原生支持根据任何数据类型的多个值或范围匹配数据包，这增强了其处理复杂过滤要求的能力。使用 nftables，您可以操作数据包中的任何字段。

在 nftables 中，集合可以是命名的或匿名的。命名的集合可被多个规则引用和动态修改。匿名集合在规则内被内联定义，并且是不可变的。集合可以包含是不同类型组合的元素，如 IP 地址和端口号对。此功能在匹配复杂标准方面提供了更大的灵活性。要管理集合，内核可以根据特定要求（性能、内存效率等）选择最合适的后端。集合也可以作为具有键值对的映射。值部分可用作数据点（写入数据包头的值），或者作为要跳到的判决或链。这可启用复杂和动态的规则行为，称为“判决映射”。

## 灵活的规则格式

nftables 规则的结构非常简单。条件和操作从左到右顺序应用。这种直观的格式简化了规则创建和故障排除。

规则中的条件在逻辑上连接（使用 AND 运算符）在一起，这意味着要使规则匹配，所有条件都必须被评估为 "true"。如果有任何条件失败，评估将移到下一个规则。

nftables 中的操作可以是最终的操作，如 drop 或 accept，这停止对数据包的进一步规则处理。非终端操作，如 counter log meta mark set 0x3，执行特定的任务（计算数据包、日志记录、设置标记等），但允许评估后续规则。

## 其他资源

- 您系统上的 `nft (8)` 手册页
- [iptables 之后是什么？当然，它的继任者是：nftables](#)
- [Firewalld：未来是 nftables](#)

### 42.2.3. 弃用的 **iptables** 框架中的概念

与主动维护的 **nftables** 框架类似，弃用的 **iptables** 框架使您能够执行各种数据包过滤任务、日志记录和审计、与 NAT 相关的配置任务等。

**iptables** 框架由多个表组成，其中每个表都为特定目的而设计：

#### **filter**

默认表确保常规数据包过滤

#### **nat**

对于网络地址转换(NAT)，包括更改数据包的源和目标地址

#### **mangle**

对于特定的数据包更改，您可以为高级路由决策对数据包标头进行修改

#### **raw**

对于需要在连接跟踪之前发生的配置

这些表作为单独的内核模块实现，其中每个表都提供一组固定的内置链，如 **INPUT**、**OUTPUT** 和 **FORWARD**。链是针对其评估数据包的规则的序列。这些将钩住内核中数据包处理流中的特定点。链在不同的表中有相同的名称，但它们的执行顺序由各自的钩子优先级决定。优先级由内核在内部管理，以确保规则以正确的顺序应用。

最初，**iptables** 被设计为处理 IPv4 流量。但是，随着 IPv6 协议的出现，需要引入 **ip6tables** 工具来提供可比较功能（如 **iptables**），并使用户能够创建和管理用于 IPv6 数据包的防火墙规则。使用相同的逻辑，创建 **arptables** 工具是为了处理地址解析协议(ARP)，开发 **ebtables** 工具是为了处理以太网桥接帧。这些工具确保您可以在各种网络协议中应用 **iptables** 的数据包过滤功能，并提供全面的网络覆盖。

要增强 **iptables** 的功能，开始开发扩展。功能扩展通常作为与用户空间动态共享对象(DSO)配对的内核模块实现。扩展引入了“匹配”和“目标”，您可以在防火墙规则中使用它们来执行更复杂的操作。扩展可以启用复杂的匹配和目标。例如，您可以匹配或操作特定的第 4 层协议标头值，执行速率限制、强制配额等。某些扩展旨在解决默认 **iptables** 语法中的限制，如“多端口”匹配扩展。此扩展允许单个规则匹配多个非连续端口，以简化规则定义，从而减少所需的单个规则的数量。

**ipset** 是 **iptables** 的一种特殊的功能扩展。这是一个内核级别的数据结构，它与 **iptables** 一起使用来创建 IP 地址、端口号和其他与网络有关的您可以匹配数据包的元素的集合。这些集合可显著简化、优化并加快编写和管理防火墙规则的过程。

## 其他资源

- [iptables \(8\) 手册页](#)

### 42.2.4. 将 **iptables** 和 **ip6tables** 规则集转换为 **nftables**

使用 **iptables-restore-translate** 和 **ip6tables-restore-translate** 实用程序将 **iptables** 和 **ip6tables** 规则集转换为 **nftables**。

## 先决条件

- 已安装 **nftables** 和 **iptables** 软件包。
- 系统配置了 **iptables** 和 **ip6tables** 规则。

## 流程

1. 将 **iptables** 和 **ip6tables** 规则写入一个文件：

```
iptables-save >/root/iptables.dump
ip6tables-save >/root/ip6tables.dump
```

2. 将转储文件转换为 **nftables** 指令：

```
iptables-restore-translate -f /root/iptables.dump > /etc/nftables/ruleset-migrated-
from-iptables.nft
ip6tables-restore-translate -f /root/ip6tables.dump > /etc/nftables/ruleset-migrated-
from-ip6tables.nft
```

3. 检查，如果需要，手动更新生成的 **nftables** 规则。

4. 要启用 **nftables** 服务来加载生成的文件，请在 **/etc/sysconfig/nftables.conf** 文件中添加以

下内容：

```
include "/etc/nftables/ruleset-migrated-from-iptables.nft"
include "/etc/nftables/ruleset-migrated-from-ip6tables.nft"
```

5.

停止并禁用 **iptables** 服务：

```
systemctl disable --now iptables
```

如果您使用自定义脚本加载 **iptables** 规则，请确保脚本不再自动启动并重新引导以刷新所有表。

6.

启用并启动 **nftables** 服务：

```
systemctl enable --now nftables
```

验证

●

显示 **nftables** 规则集：

```
nft list ruleset
```

其它资源

●

[系统引导时自动载入 \*\*nftables\*\* 规则](#)

#### 42.2.5. 将单个 **iptables** 和 **ip6tables** 规则转换为 **nftables**

Red Hat Enterprise Linux 提供了 **iptables-translate** 和 **ip6tables-translate** 工具来将 **iptables** 或 **ip6tables** 规则转换为与 **nftables** 相等的规则。

前提条件

●

已安装 **nftables** 软件包。

## 流程

- 使用 **iptables-translate** 或 **ip6tables-translate** 程序而不是 **iptables** 或 **ip6tables** 显示对应的 **nftables** 规则，例如：

```
iptables-translate -A INPUT -s 192.0.2.0/24 -j ACCEPT
nft add rule ip filter INPUT ip saddr 192.0.2.0/24 counter accept
```

请注意，一些扩展可能缺少响应的转换支持。在这些情况下，实用程序会输出以 # 符号为前缀的未转换规则，例如：

```
iptables-translate -A INPUT -j CHECKSUM --checksum-fill
nft # -A INPUT -j CHECKSUM --checksum-fill
```

## 其它资源

- **iptables-translate --help**

### 42.2.6. 常见的 **iptables** 和 **nftables** 命令的比较

以下是常见 **iptables** 和 **nftables** 命令的比较：

- 列出所有规则：

<b>iptables</b>	<b>nftables</b>
<b>iptables-save</b>	<b>nft list ruleset</b>

- 列出某个表和链：

<b>iptables</b>	<b>nftables</b>
<b>iptables -L</b>	<b>nft list table ip filter</b>
<b>iptables -L INPUT</b>	<b>nft list chain ip filter INPUT</b>
<b>iptables -t nat -L PREROUTING</b>	<b>nft list chain ip nat PREROUTING</b>

**nft 命令不会预先创建表和链。只有当用户手动创建它们时它们才会存在。**

列出 **firewalld** 生成的规则：

```
nft list table inet firewalld
nft list table ip firewalld
nft list table ip6 firewalld
```

### 42.3. 使用 NFTABLES 配置 NAT

有了 **nftables**，您就可以配置以下网络地址转换(NAT)类型：

- 伪装
- 源 NAT (SNAT)
- 目标 NAT (DNAT)
- 重定向



重要

您只能在 **iifname** 和 **oifname** 参数中使用实际的接口名称，不支持替代名称 (**altname**)。

#### 42.3.1. NAT 类型

这些是不同的网络地址转换 (NAT) 类型：

##### 伪装和源 NAT (SNAT)

使用以上 NAT 类型之一更改数据包的源 IP 地址。例如，互联网服务提供商(ISP)不会路由私有 IP 范围，如 10.0.0.0/8。如果您在网络中使用私有 IP 范围，用户应该能够访问互联网上的服务器，请将

来自这些范围的数据包的源 IP 地址映射为公共 IP 地址。

伪装和 SNAT 相互类似。不同之处是：

- 伪装自动使用传出接口的 IP 地址。因此，如果传出接口使用了动态 IP 地址，则使用伪装。
- SNAT 将数据包的源 IP 地址设置为指定的 IP 地址，且不会动态查找传出接口的 IP 地址。因此，SNAT 要比伪装更快。如果传出接口使用了固定 IP 地址，则使用 SNAT。

## 目标 NAT (DNAT)

使用此 NAT 类型重写传入数据包的目标地址和端口。例如，如果您的 Web 服务器使用来自私有 IP 范围的 IP 地址，因此无法直接从互联网访问，您可以在路由器上设置 DNAT 规则，来将传入的流量重定向到此服务器。

### 重定向

这个类型是 IDT 的特殊示例，它根据链 hook 将数据包重定向到本地机器。例如，如果服务运行在与其标准端口不同的端口上，您可以将传入的流量从标准端口重定向到此特定端口。

#### 42.3.2. 使用 nftables 配置伪装

伪装使路由器动态地更改通过接口到接口 IP 地址发送的数据包的源 IP。这意味着，如果接口被分配了一个新 IP，nftables 会在替换源 IP 时自动使用新的 IP。

将通过 ens3 接口离开主机的数据包源 IP 替换为 ens3 上设置的 IP。

### 流程

1.

创建一个表：

```
nft add table nat
```

2.

向表中添加 prerouting 和 postrouting 链：

```
nft add chain nat postrouting { type nat hook postrouting priority 100 }; }
```



### 重要

即使您没有向 **prerouting** 链中添加规则，**nftables** 框架也会要求此链与传入的数据包回复匹配。

请注意，您必须将 **--** 选项传递给 **nft** 命令，以防止 **shell** 将负优先级值解释为 **nft** 命令的选项。

3.

向 **postrouting** 链中添加一条规则，来匹配 **ens3** 接口上传出的数据包：

```
nft add rule nat postrouting oifname "ens3" masquerade
```

### 42.3.3. 使用 **nftables** 配置源 NAT

在路由器中，源 NAT（SNAT）可让您将通过接口发送的数据包 IP 改为专门的 IP 地址。然后，路由器会替换传出数据包的源 IP。

### 流程

1.

创建一个表：

```
nft add table nat
```

2.

向表中添加 **prerouting** 和 **postrouting** 链：

```
nft add chain nat postrouting { type nat hook postrouting priority 100 }; }
```



### 重要

即使您没有向 **postrouting** 链添加规则，**nftables** 框架也会要求此链与传出数据包回复相匹配。

请注意，您必须将 **--** 选项传递给 **nft** 命令，以防止 **shell** 将负优先级值解释为 **nft** 命令的选

项。

3.

向 **postROUTING** 链中添加一条规则，该规则将使用 192.0.2.1 替换通过 **ens3** 的传出数据包的源 IP：

```
nft add rule nat postROUTING oifname "ens3" snat to 192.0.2.1
```

#### 42.3.4. 使用 nftables 配置目标 NAT

目标 NAT (DNAT) 可让您将路由器上的流量重定向到无法直接从互联网访问的主机。

例如，有了 DNAT，路由器将发送给端口 80 和 443 的传入流量重定向到 IP 地址为 192.0.2.1 的 Web 服务器。

#### 流程

1.

创建一个表：

```
nft add table nat
```

2.

向表中添加 **prerouting** 和 **postROUTING** 链：

```
nft -- add chain nat prerouting { type nat hook prerouting priority -100 \; }
nft add chain nat postROUTING { type nat hook postROUTING priority 100 \; }
```



重要

即使您没有向 **postROUTING** 链添加规则，nftables 框架也会要求此链与传出数据包回复相匹配。

请注意，您必须将 **--** 选项传递给 **nft** 命令，以防止 shell 将负优先级值解释为 **nft** 命令的选项。

3.

向 **prerouting** 链中添加一条规则，该规则将路由器的 **ens3** 接口上端口 80 和 443 的传入流量重定向到 IP 地址为 192.0.2.1 的 web 服务器：

```
nft add rule nat prerouting iifname ens3 tcp dport { 80, 443 } dnat to 192.0.2.1
```

4.

根据您的环境，添加 SNAT 或伪装规则，将从 Web 服务器返回的数据包的源地址改为发送者：

a.

如果 ens3 接口使用动态 IP 地址，请添加一条伪装规则：

```
nft add rule nat postrouting oifname "ens3" masquerade
```

b.

如果 ens3 接口使用静态 IP 地址，请添加一条 SNAT 规则。例如，如果 ens3 使用 198.51.100.1 IP 地址：

```
nft add rule nat postrouting oifname "ens3" snat to 198.51.100.1
```

5.

启用数据包转发：

```
echo "net.ipv4.ip_forward=1" > /etc/sysctl.d/95-IPv4-forwarding.conf
sysctl -p /etc/sysctl.d/95-IPv4-forwarding.conf
```

## 其它资源



[NAT 类型](#)

### 42.3.5. 使用 nftables 配置重定向

重定向 功能是目标网络地址转换(DNAT)的一种特殊情况，它根据链 hook 将数据包重定向到本地计算机。

例如，您可以将发送到本地主机端口 22 的流量重定向到端口 2222。

## 流程

1.

创建一个表：



```
nft add table nat
```

2.

向表中添加 `prerouting` 链：

```
nft -- add chain nat prerouting { type nat hook prerouting priority -100 \; }
```

请注意，您必须将 `--` 选项传递给 `nft` 命令，以防止 shell 将负优先级值解释为 `nft` 命令的选项。

3.

向 `prerouting` 链中添加一条规则，其将端口 22 上的传入流量重定向到端口 2222：

```
nft add rule nat prerouting tcp dport 22 redirect to 2222
```

## 其它资源



[NAT 类型](#)

### 42.4. 编写和执行 NFTABLES 脚本

使用 `nftables` 框架的主要优点是脚本的执行是原子的。这意味着，系统会应用整个脚本，或者在出现错误时防止执行。这样可保证防火墙始终处于一致状态。

另外，使用 `nftables` 脚本环境时，您可以：



[添加评论](#)



[定义变量](#)



[包括其他规则集文件](#)

安装 `nftables` 软件包时，Red Hat Enterprise Linux 会在 `/etc/nftables/` 目录中自动创建 `*.nft` 脚本。这些脚本包含为不同目的创建表和空链的命令。

#### 42.4.1. 支持的 nftables 脚本格式

您可以使用以下格式在 nftables 脚本环境中编写脚本：

- 与 nft list ruleset 命令相同的格式显示规则集：

```
#!/usr/sbin/nft -f

Flush the rule set
flush ruleset

table inet example_table {
 chain example_chain {
 # Chain for incoming packets that drops all packets that
 # are not explicitly allowed by any rule in this chain
 type filter hook input priority 0; policy drop;

 # Accept connections to port 22 (ssh)
 tcp dport ssh accept
 }
}
```

- 与 nft 命令的语法相同：

```
#!/usr/sbin/nft -f

Flush the rule set
flush ruleset

Create a table
add table inet example_table

Create a chain for incoming packets that drops all packets
that are not explicitly allowed by any rule in this chain
add chain inet example_table example_chain { type filter hook input priority 0 ; policy drop ; }

Add a rule that accepts connections to port 22 (ssh)
add rule inet example_table example_chain tcp dport ssh accept
```

#### 42.4.2. 运行 nftables 脚本

您可以通过将脚本传递给 nft 实用程序或直接执行脚本来运行 nftables 脚本。

流程

- 要通过将其传给 `nft` 工具来运行 `nftables` 脚本，请输入：

```
nft -f /etc/nftables/<example_firewall_script>.nft
```

- 要直接运行 `nftables` 脚本：

a.

在进行这个时间时：

i.

确保脚本以以下 shebang 序列开头：

```
#!/usr/sbin/nft -f
```



重要

如果省略了 `-f` 参数，`nft` 工具不会读取脚本，并显示 `Error: syntax error, unexpected newline, expecting string.`

ii.

可选：将脚本的所有者设为 `root`：

```
chown root /etc/nftables/<example_firewall_script>.nft
```

iii.

使脚本可以被其所有者执行：

```
chmod u+x /etc/nftables/<example_firewall_script>.nft
```

b.

运行脚本：

```
/etc/nftables/<example_firewall_script>.nft
```

如果没有输出结果，系统将成功执行该脚本。



## 重要

即使 nft 成功地执行了脚本，在脚本中错误放置的规则、缺失的参数或其他问题都可能导致防火墙的行为不符合预期。

## 其他资源

- 您系统上的 **chown (1)** 和 **chmod (1)** 手册页
- 系统引导时自动载入 **nftables** 规则

### 42.4.3. 使用 nftables 脚本中的注释

**nftables** 脚本环境将 # 字符右侧的所有内容解释为行尾。

注释可在行首或命令旁边开始：

```
...
Flush the rule set
flush ruleset

add table inet example_table # Create a table
...
```

### 42.4.4. 使用 nftables 脚本中的变量

要在 **nftables** 脚本中定义一个变量，请使用 **define** 关键字。您可以在变量中存储单个值和匿名集合。对于更复杂的场景，请使用 **set** 或 **verdict** 映射。

#### 只有一个值的变量

以下示例定义了一个名为 **INET\_DEV** 的变量，其值为 **enp1s0**：

```
define INET_DEV = enp1s0
```

您可以通过输入 \$ 符号后再输入变量名称来使用脚本中的变量：

```
...
add rule inet example_table example_chain iifname $INET_DEV tcp dport ssh accept
...
```

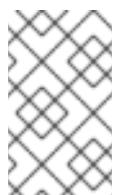
## 包含匿名集合的变量

以下示例定义了一个包含匿名集合的变量：

```
define DNS_SERVERS = { 192.0.2.1, 192.0.2.2 }
```

您可以通过在 \$ 符号后跟变量名称来在脚本中使用变量：

```
add rule inet example_table example_chain ip daddr $DNS_SERVERS accept
```



### 注意

当您在规则中使用大括号时具有特殊的语义，因为它们表示变量代表一个集合。

## 其他资源

- [使用 nftables 命令中的设置](#)
- [在 nftables 命令中使用 verdict 映射](#)

### 42.4.5. 在 nftables 脚本中包含文件

在 nftables 脚本环境中，您可以使用 `include` 语句包含其他脚本。

如果您只指定没有绝对或相对路径的文件名，nftables 包含了默认搜索路径（设置为 Red Hat Enterprise Linux 上的 /etc）。

#### 例 42.1. 包含默认搜索目录中的文件

从默认搜索目录中包含一个文件：

```
include "example.nft"
```

#### 例 42.2. 包含目录中的所有 \*.nft 文件

要包括以 `*.nft` 结尾的所有文件，它们存储在 `/etc/nftables/rulesets/` 目录中：

```
include "/etc/nftables/rulesets/*.nft"
```

请注意，`include` 语句不匹配以点开头的文件。

### 其他资源

- 您系统上 `nft (8)` 手册页中的 **Include files** 部分

#### 42.4.6. 系统引导时自动载入 nftables 规则

`nftables` `systemd` 服务加载包含在 `/etc/sysconfig/nftables.conf` 文件中的防火墙脚本。

### 先决条件

- `nftables` 脚本存储在 `/etc/nftables/` 目录中。

### 流程

1.

编辑 `/etc/sysconfig/nftables.conf` 文件。

- 如果您使用 `nftables` 软件包的安装修改了在 `/etc/nftables/` 中创建的 `*.nft` 脚本，请取消对这些脚本的 `include` 语句的注释。
- 如果您编写了新脚本，请添加 `include` 语句以包含这些脚本。例如，要在 `nftables` 服务启动时加载 `/etc/nftables/example.nft` 脚本，请添加：

```
include "/etc/nftables/_example_.nft"
```

2.

可选：启动 **nftables** 服务来加载防火墙规则，而无需重启系统：

```
systemctl start nftables
```

3.

启用 **nftables** 服务。

```
systemctl enable nftables
```

## 其它资源

- 

[支持的 nftables 脚本格式](#)

## 42.5. 使用 NFTABLES 命令中的设置

**nftables** 框架原生支持集合。您可以使用一个集合，例如，规则匹配多个 IP 地址、端口号、接口或其他匹配标准。

### 42.5.1. 在 nftables 中使用匿名集合

匿名集合包含用逗号分开的值，如 { 22、80、443 }，它们直接在规则中使用。您还可以将匿名集合用于 IP 地址以及任何其他匹配标准。

匿名集合的缺陷是，如果要更改集合，则需要替换规则。对于动态解决方案，使用命名集合，如 [在 nftables 中使用命名集合](#) 中所述。

## 先决条件

- 

inet 系列中的 **example\_chain** 链和 **example\_table** 表存在。

## 流程

1.

例如，要向 **example\_table** 中的 **example\_chain** 添加一条规则，其允许传入流量到端口 22、80 和 443：

```
nft add rule inet example_table example_chain tcp dport { 22, 80, 443 } accept
```

2.

可选：在 `example_table` 中显示所有链及其规则：

```
nft list table inet example_table
table inet example_table {
 chain example_chain {
 type filter hook input priority filter; policy accept;
 tcp dport { ssh, http, https } accept
 }
}
```

#### 42.5.2. 在 nftables 中使用命名集

**nftables** 框架支持可变命名集合。命名集是一个列表或一组元素，您可以在表中的多个规则中使用。匿名集合的另外一个好处在于，您可以更新命名的集合而不必替换使用集合的规则。

当您创建一个命名集时，必须指定集合包含的元素类型。您可以设置以下类型：

- 包含 IPv4 地址或范围的集合的 `ipv4_addr`, 如 `192.0.2.1` 或 `192.0.2.0/24`。
- 包含 IPv6 地址或范围的集合的 `ipv6_addr`, 如 `2001:db8:1::1` 或 `2001:db8:1::1/64`。
- 包含介质访问控制(MAC)地址列表的集合的 `ether_addr`, 如 `52:54:00:6b:66:42`。
- 包含互联网协议类型列表的集合的 `inet_proto`, 如 `tcp`。
- 包含互联网服务列表的集合的 `inet_service`, 如 `ssh`。
- 包含数据包标记列表的集合的 `mark`。数据包标记可以是任意正 32 位整数值(0 到 2147483647)。

#### 先决条件

- `example_chain` 链和 `example_table` 表存在。

## 流程

1.

创建一个空集。以下示例为 IPv4 地址创建了一个集合：

- 要创建可存储多个独立 IPv4 地址的集合：

```
nft add set inet example_table example_set { type ipv4_addr \; }
```

- 要创建可存储 IPv4 地址范围的集合：

```
nft add set inet example_table example_set { type ipv4_addr \; flags interval \; }
```



重要

要防止 shell 将分号解释为命令结尾，您必须使用反斜杠转义分号。

2.

可选：创建使用集合的规则。例如，以下命令向 example\_table 中的 example\_chain 中添加一条规则，该规则将丢弃 example\_set 中来自 IPv4 地址的所有数据包。

```
nft add rule inet example_table example_chain ip saddr @example_set drop
```

由于 example\_set 仍为空，所以该规则目前不起作用。

3.

向 example\_set 中添加 IPv4 地址：

- 如果您创建存储单个 IPv4 地址的集合，请输入：

```
nft add element inet example_table example_set { 192.0.2.1, 192.0.2.2 }
```

- 如果您创建存储 IPv4 范围的集合，请输入：

```
nft add element inet example_table example_set { 192.0.2.0-192.0.2.255 }
```

当您指定 IP 地址范围时，您可以使用无类别域间路由(CIDR)表示法，如上例中的 192.0.2.0/24。

#### 42.5.3. 使用动态集添加来自数据包路径的条目

**nftables** 框架中的动态设置允许自动添加来自数据包数据的元素。例如，IP 地址、目标端口、MAC 地址等。通过此功能，您可以实时收集这些元素，并使用它们创建拒绝列表、禁止列表等，以便您能够立即响应安全威胁。

##### 先决条件

- inet 系列中的 `example_chain` 链和 `example_table` 表存在。

##### 流程

1. 创建一个空集。以下示例为 IPv4 地址创建了一个集合：

- 要创建可存储多个独立 IPv4 地址的集合：

```
nft add set inet example_table example_set { type ipv4_addr \; }
```

- 要创建可存储 IPv4 地址范围的集合：

```
nft add set inet example_table example_set { type ipv4_addr \; flags interval \; }
```



##### 重要

要防止 shell 将分号解释为命令结尾，您必须使用反斜杠转义分号。

2. 创建一条规则，将传入数据包的源 IPv4 地址添加到 `example_set` 组中：

```
nft add rule inet example_table example_chain set add ip saddr @example_set
```

命令在 `example_chain` 规则链中创建了一个新规则，并使用 `example_table` 将数据包的源 IPv4 地址动态添加到 `example_set` 中。

## 验证

- 确保添加了规则：

```
nft list ruleset
...
table ip example_table {
 set example_set {
 type ipv4_addr
 elements = { 192.0.2.250, 192.0.2.251 }
 }

 chain example_chain {
 type filter hook input priority 0
 add @example_set { ip saddr }
 }
}
```

命令显示 nftables 中当前载入的整个规则集。它显示 IP 正在主动触发规则，并且使用相关地址更新 example\_set。

## 后续步骤

有动态 IP 集后，您可以将它用于各种安全性、过滤和流量控制目的。例如：

- 块、限制或记录网络流量
- 与允许列表结合使用，以避免禁止可信用户
- 使用自动超时来防止过度阻塞

### 42.5.4. 其他资源

- 您系统上 nft (8) 手册页中的 Sets 部分

## 42.6. 在 NFTABLES 命令中使用判决映射

判决映射（也称为字典），使 nft 能够通过将匹配条件映射到某个操作来根据数据包信息执行操作。

#### 42.6.1. 在 nftables 中使用匿名映射

匿名映射是直接在规则中使用的 { *match\_criteria* : *action* } 语句。这个语句可以包含多个用逗号分开的映射。

匿名映射的缺点是，如果要修改映射，则必须替换规则。对于动态解决方案，请使用命名映射，如 [在 nftables 中使用命名映射](#) 中所述。

例如，您可以使用匿名映射将 IPv4 和 IPv6 协议的 TCP 和 UDP 数据包路由到不同的链，以分别计算传入的 TCP 和 UDP 数据包。

#### 流程

1.

创建新表：

```
nft add table inet example_table
```

2.

在 example\_table 中创建 tcp\_packets 链：

```
nft add chain inet example_table tcp_packets
```

3.

向统计此链中流量的 tcp\_packets 中添加一条规则：

```
nft add rule inet example_table tcp_packets counter
```

4.

在 example\_table 中创建 udp\_packets 链

```
nft add chain inet example_table udp_packets
```

5.

向统计此链中流量的 udp\_packets 中添加一条规则：

```
nft add rule inet example_table udp_packets counter
```

6.

为传入的流量创建一个链。例如，要在过滤传入的流量的 `example_table` 中创建一个名为 `incoming_traffic` 的链：

```
nft add chain inet example_table incoming_traffic { type filter hook input priority 0 \;
}
```

7.

添加一条带有到 `incoming_traffic` 匿名映射的规则：

```
nft add rule inet example_table incoming_traffic ip protocol vmap { tcp : jump
tcp_packets, udp : jump udp_packets }
```

匿名映射区分数据包，并根据它们的协议将它们发送到不同的计数链。

8.

要列出流量计数器，请显示 `example_table`：

```
nft list table inet example_table
table inet example_table {
 chain tcp_packets {
 counter packets 36379 bytes 2103816
 }

 chain udp_packets {
 counter packets 10 bytes 1559
 }

 chain incoming_traffic {
 type filter hook input priority filter; policy accept;
 ip protocol vmap { tcp : jump tcp_packets, udp : jump udp_packets }
 }
}
```

`tcp_packets` 和 `udp_packets` 链中的计数器显示两者接收的数据包和字节数。

#### 42.6.2. 在 nftables 中使用命名映射

`nftables` 框架支持命名映射。您可以在表中的多个规则中使用这些映射。匿名映射的另一个好处在于，您可以更新命名映射而不比替换使用它的规则。

在创建命名映射时，您必须指定元素的类型：

- 匹配部分包含 IPv4 地址的映射的 `ipv4_addr`, 如 `192.0.2.1`。
- 匹配部分包含 IPv6 地址的映射的 `ipv6_addr`, 如 `2001:db8:1::1`。
- 匹配部分包含介质访问控制(MAC)地址的映射的 `ether_addr`, 如 `52:54:00:6b:66:42`。
- 匹配部分包含互联网协议类型的映射的 `inet_proto`, 如 `tcp`。
- 匹配部分包含互联网服务名称端口号的映射的 `inet_service`, 如 `ssh` 或 `22`。
- 匹配部分包含数据包的映射的 `mark`。数据包标记可以是任意正 32 位整数值(0 到 `2147483647`)。
- 匹配部分包含计数器值的映射的 `counter`。计数器值可以是任意正 64 位整数值。
- 匹配部分包含配额值的映射的 `quota`。配额值可以是任意正 64 位整数值。

例如，您可以根据其源 IP 地址允许或丢弃传入的数据包。使用命名映射时，您只需要一条规则来配置这种场景，而 IP 地址和操作被动态存储在映射中。

## 流程

1.

创建表。例如，要创建一个处理 IPv4 数据包的、名为 `example_table` 的表：

```
nft add table ip example_table
```

2.

创建链。例如，要在 `example_table` 中创建一个名为 `example_chain` 的链：

```
nft add chain ip example_table example_chain { type filter hook input priority 0 \; }
```



重要

要防止 shell 将分号解释为命令结尾，您必须使用反斜杠转义分号。

3.

创建一个空的映射。例如，要为 IPv4 地址创建映射：

```
nft add map ip example_table example_map { type ipv4_addr : verdict \; }
```

4.

创建使用该映射的规则。例如，以下命令向 example\_table 中的 example\_chain 添加了一条规则，该规则将操作应用到在 example\_map 中定义的 IPv4 地址：

```
nft add rule example_table example_chain ip saddr vmap @example_map
```

5.

向 example\_map 添加 IPv4 地址和相应的操作：

```
nft add element ip example_table example_map { 192.0.2.1 : accept, 192.0.2.2 : drop }
```

这个示例定义了 IPv4 地址到操作的映射。与以上创建的规则相结合，防火墙接受来自 192.0.2.1 的数据包，丢弃来自 192.0.2.2 的数据包。

6.

可选：通过添加另一个 IP 地址和 action 语句来增强映射：

```
nft add element ip example_table example_map { 192.0.2.3 : accept }
```

7.

可选：从映射中删除条目：

```
nft delete element ip example_table example_map { 192.0.2.1 }
```

8.

可选：显示规则集：

```
nft list ruleset
table ip example_table {
 map example_map {
```

```

type ipv4_addr : verdict
elements = { 192.0.2.2 : drop, 192.0.2.3 : accept }
}

chain example_chain {
 type filter hook input priority filter; policy accept;
 ip saddr vmap @example_map
}
}

```

#### 42.6.3. 其他资源

- 您系统上 **nft (8)** 手册页中的 **Maps** 部分

### 42.7. 示例：使用 NFTABLES 脚本保护 LAN 和 DMZ

使用 RHEL 路由器上的 **nftables** 框架来编写和安装防火墙脚本，来保护内部 LAN 中的网络客户端和 DMZ 中的 Web 服务器免受来自互联网和其他网络的未授权访问。



#### 重要

这个示例仅用于演示目的，描述了一个具有特定要求的场景。

防火墙脚本高度依赖网络基础架构和安全要求。当您为自己的环境编写脚本时，请使用此示例了解 **nftables** 防火墙的概念。

#### 42.7.1. 网络条件

本例中的网络具有以下条件：

- 路由器连接到以下网络：
  - 互联网通过接口 **enp1s0**
  - 内部 LAN 通过接口 **enp7s0**

- DMZ 通过 `enp8s0`
- 路由器的互联网接口分配了静态 IPv4 地址(203.0.113.1)和 IPv6 地址(2001:db8:a::1)。
- 内部 LAN 中的客户端仅使用范围 10.0.0.0/24 中的专用 IPv4 地址。因此，从 LAN 到互联网的流量需要源网络地址转换(SNAT)。
- 内部 LAN 中的管理员 PC 使用 IP 地址 10.0.0.100 和 10.0.0.200。
- DMZ 使用范围 198.51.100.0/24 和 2001:db8:b::/56 中的公共 IP 地址。
- DMZ 中的 Web 服务器使用 IP 地址 198.51.100.5 和 2001:db8:b::5。
- 路由器为 LAN 和 DMZ 中的主机充当缓存 DNS 服务器。

#### 42.7.2. 防火墙脚本的安全要求

以下是示例网络中 `nftables` 防火墙的要求：

- 路由器必须能够：
  - 递归解析 DNS 查询。
  - 在回环接口上执行所有连接。
- 内部 LAN 中的客户端必须能够：
  - 查询运行在路由器上的缓存 DNS 服务器。

- 访问 DMZ 中的 HTTPS 服务器。
- 访问互联网上的任何 HTTPS 服务器。
- 管理员的 PC 必须能够使用 SSH 访问 DMZ 中的路由器以及每个服务器。
- DMZ 中的 Web 服务器必须能够：
  - 查询运行在路由器上的缓存 DNS 服务器。
  - 访问互联网上的 HTTPS 服务器以下载更新。
- 互联网上的主机必须能够：
  - 访问 DMZ 中的 HTTPS 服务器。
- 另外，存在以下安全要求：
  - 应丢弃未明确允许的连接尝试。
  - 应记录丢弃的数据包。

#### 42.7.3. 配置将丢弃的数据包记录到文件

默认情况下，`systemd` 会将内核消息如，丢弃的数据包，记录到日志中。另外，您可以配置 `rsyslog` 服务，来将此类条目记录到单独的文件中。为确保日志文件不会无限增长，请配置轮转策略。

##### 先决条件

- **rsyslog** 软件包已安装。
- **rsyslog** 服务正在运行。

## 流程

1. 使用以下内容创建 `/etc/rsyslog.d/nftables.conf` 文件：

```
:msg, startswith, "nft drop" -/var/log/nftables.log
& stop
```

使用这个配置，**rsyslog** 服务会将丢弃的数据包记录到 `/var/log/nftables.log` 文件，而不是 `/var/log/messages`。

2. 重启 **rsyslog** 服务：

```
systemctl restart rsyslog
```

3. 使用以下内容创建 `/etc/logrotate.d/nftables` 文件，以便在大小超过 10 MB 时轮转 `/var/log/nftables.log`：

```
/var/log/nftables.log {
 size +10M
 maxage 30
 sharedscripts
 postrotate
 /usr/bin/systemctl kill -s HUP rsyslog.service >/dev/null 2>&1 || true
 endscript
}
```

`maxage 30` 设置定义了 `logrotate` 在下一次轮转操作过程中删除超过 30 天的轮转日志。

## 其他资源

- 您系统上的 `rsyslog.conf (5)` 和 `logrotate (8)` 手册页

### 42.7.4. 编写并激活 nftables 脚本

本例是运行在 RHEL 路由器上的一个 nftables 防火墙脚本，其保护内部 LAN 中的客户端以及 DMZ 中的 Web 服务器。有关示例中使用的防火墙的网络和要求的详情，请参阅 [网络条件](#) 和 [对防火墙脚本的安全要求](#)。



### 警告

此 nftables 防火墙脚本仅用于演示目的。不要在未经调整为适合您环境和安全要求的情况下使用它。

## 前提条件

- 网络已配置，如 [网络状况](#) 中所述。

## 流程

1. 使用以下内容创建 /etc/nftables/firewall.nft 脚本：

```
Remove all rules
flush ruleset

Table for both IPv4 and IPv6 rules
table inet nftables_svc {

 # Define variables for the interface name
 define INET_DEV = enp1s0
 define LAN_DEV = enp7s0
 define DMZ_DEV = enp8s0

 # Set with the IPv4 addresses of admin PCs
 set admin_pc_ipv4 {
 type ipv4_addr
 elements = { 10.0.0.100, 10.0.0.200 }
 }

 # Chain for incoming traffic. Default policy: drop
 chain INPUT {
 type filter hook input priority filter
 policy drop
```

```

Accept packets in established and related state, drop invalid packets
ct state vmap { established:accept, related:accept, invalid:drop }

Accept incoming traffic on loopback interface
iifname lo accept

Allow request from LAN and DMZ to local DNS server
iifname { $LAN_DEV, $DMZ_DEV } meta l4proto { tcp, udp } th dport 53 accept

Allow admins PCs to access the router using SSH
iifname $LAN_DEV ip saddr @admin_pc_ipv4 tcp dport 22 accept

Last action: Log blocked packets
(packets that were not accepted in previous rules in this chain)
log prefix "nft drop IN :"
}

Chain for outgoing traffic. Default policy: drop
chain OUTPUT {
 type filter hook output priority filter
 policy drop

Accept packets in established and related state, drop invalid packets
ct state vmap { established:accept, related:accept, invalid:drop }

Accept outgoing traffic on loopback interface
oifname lo accept

Allow local DNS server to recursively resolve queries
oifname $INET_DEV meta l4proto { tcp, udp } th dport 53 accept

Last action: Log blocked packets
log prefix "nft drop OUT :"
}

Chain for forwarding traffic. Default policy: drop
chain FORWARD {
 type filter hook forward priority filter
 policy drop

Accept packets in established and related state, drop invalid packets
ct state vmap { established:accept, related:accept, invalid:drop }

IPv4 access from LAN and internet to the HTTPS server in the DMZ
iifname { $LAN_DEV, $INET_DEV } oifname $DMZ_DEV ip daddr 198.51.100.5 tcp
dport 443 accept

IPv6 access from internet to the HTTPS server in the DMZ
iifname $INET_DEV oifname $DMZ_DEV ip6 daddr 2001:db8:b::5 tcp dport 443
accept

Access from LAN and DMZ to HTTPS servers on the internet
iifname { $LAN_DEV, $DMZ_DEV } oifname $INET_DEV tcp dport 443 accept

```

```

Last action: Log blocked packets
log prefix "nft drop FWD: "
}

Postrouting chain to handle SNAT
chain postROUTING {
 type nat hook postROUTING priority srcnat; policy accept;

 # SNAT for IPv4 traffic from LAN to internet
 iifname $LAN_DEV oifname $INET_DEV snat ip to 203.0.113.1
}
}

```

2.

在 /etc/sysconfig/nftables.conf 文件中包括 /etc/nftables/firewall.nft 脚本：

```
include "/etc/nftables/firewall.nft"
```

3.

启用 IPv4 转发：

```
echo "net.ipv4.ip_forward=1" > /etc/sysctl.d/95-IPv4-forwarding.conf
sysctl -p /etc/sysctl.d/95-IPv4-forwarding.conf
```

4.

启用并启动 nftables 服务：

```
systemctl enable --now nftables
```

## 验证

1.

可选：验证 nftables 规则集：

```
nft list ruleset
...
```

2.

尝试执行防火墙阻止的访问。例如，尝试使用 SSH 从 DMZ 访问路由器：

```
ssh router.example.com
ssh: connect to host router.example.com port 22: Network is unreachable
```

3. 根据您的日志记录设置，搜索：

- 阻塞的数据包的 systemd 日志：

```
journalctl -k -g "nft drop"
Oct 14 17:27:18 router kernel: nft drop IN :IN=enp8s0 OUT= MAC=...
SRC=198.51.100.5 DST=198.51.100.1 ... PROTO=TCP SPT=40464 DPT=22 ... SYN ...
```

- 阻塞的数据包的 /var/log/nftables.log 文件：

```
Oct 14 17:27:18 router kernel: nft drop IN :IN=enp8s0 OUT= MAC=...
SRC=198.51.100.5 DST=198.51.100.1 ... PROTO=TCP SPT=40464 DPT=22 ... SYN ...
```

## 42.8. 使用 NFTABLES 限制连接的数量

您可以使用 nftables 来限制连接数或限制到建立给定数量连接的块 IP 地址，以防止它们使用太多的系统资源。

### 42.8.1. 使用 nftables 限制连接的数量

通过使用 nft 工具的 ct count 参数，您可以限制每个 IP 地址的同时连接的数量。例如，您可以使用此功能配置每个源 IP 地址只能建立两个到主机的并行 SSH 连接。

#### 流程

1. 创建带有 inet 地址系列的 filter 表：

```
nft add table inet filter
```

2. 将 input 链添加到 inet filter 表中：

```
nft add chain inet filter input { type filter hook input priority 0 \; }
```

3. 为 IPv4 地址创建动态集合：

```
nft add set inet filter limit-ssh { type ipv4_addr; flags dynamic \;}
```

4.

在 **input** 链中添加一条规则，该规则只允许从 IPv4 地址到 SSH 端口(22)的两个同时的传入连接，并拒绝来自同一 IP 的所有进一步的连接：

```
nft add rule inet filter input tcp dport ssh ct state new add @limit-ssh { ip saddr ct count over 2 } counter reject
```

## 验证

1.

建立从同一 IP 地址到主机的两个以上的同时的 SSH 连接。如果已经建立了两个连接，则 **nftables** 会拒绝到 SSH 端口的连接。

2.

显示 **limit-ssh** 动态集：

```
nft list set inet filter limit-ssh
table inet filter {
 set limit-ssh {
 type ipv4_addr
 size 65535
 flags dynamic
 elements = { 192.0.2.1 ct count over 2 , 192.0.2.2 ct count over 2 }
 }
}
```

**elements** 条目显示当前与该规则匹配的地址。在本例中，**elements** 列出了与 SSH 端口有活动连接的 IP 地址。请注意，输出不会显示活跃连接的数量，或者连接是否被拒绝。

### 42.8.2. 在一分钟内尝试超过十个进入的 TCP 连接的 IP 地址

您可以临时阻止在一分钟内建立十个 IPv4 TCP 连接的主机。

## 流程

1.

创建具有 **ip** 地址系列的 **filter** 表：

```
nft add table ip filter
```

2.

向 **filter** 表中添加 **input** 链：

```
nft add chain ip filter input { type filter hook input priority 0 \; }
```

3.

添加一条规则，其丢弃来自源地址的所有数据包，并尝试在一分钟内建立十个 TCP 连接：

```
nft add rule ip filter input ip protocol tcp ct state new, untracked meter ratemeter { ip saddr timeout 5m limit rate over 10/minute } drop
```

**timeout 5m** 参数定义 **nftables** 在五分钟后自动删除条目，以防止计量被过时的条目填满。

## 验证



要显示 **meter** 的内容，请输入：

```
nft list meter ip filter ratemeter
table ip filter {
 meter ratemeter {
 type ipv4_addr
 size 65535
 flags dynamic,timeout
 elements = { 192.0.2.1 limit rate over 10/minute timeout 5m expires 4m58s224ms }
 }
}
```

## 42.9. 调试 NFTABLES 规则

**nftables** 框架为管理员提供了不同的选项来调试规则，以及数据包是否匹配。

### 42.9.1. 创建带有计数器的规则

在识别规则是否匹配时，可以使用计数器。



有关在现有规则中添加计数器的流程的更多信息，请参阅 [向现有规则中添加计数器](#)。

## 先决条件

- 您要添加该规则的链已存在。

## 流程

1. 向链中添加一条带有 `counter` 参数的新规则。以下示例添加一个带有计数器的规则，它允许端口 22 上的 TCP 流量，并计算与这个规则匹配的数据包和网络数据的数量：

```
nft add rule inet example_table example_chain tcp dport 22 counter accept
```

2. 显示计数器值：

```
nft list ruleset
table inet example_table {
 chain example_chain {
 type filter hook input priority filter; policy accept;
 tcp dport ssh counter packets 6872 bytes 105448565 accept
 }
}
```

### 42.9.2. 在现有规则中添加计数器

在识别规则是否匹配时，可以使用计数器。

- 有关使用计数器添加新规则的流程的更多信息，请参阅 [创建带有计数器的规则](#)。

## 先决条件

- 您要添加计数器的规则已存在。

## 流程

1. 在链中显示规则及其句柄：

```
nft --handle list chain inet example_table example_chain
table inet example_table {
 chain example_chain { # handle 1
 type filter hook input priority filter; policy accept;
 tcp dport ssh accept # handle 4
 }
}
```

```
 }
```

2.

通过使用 `counter` 参数替换规则来添加计数器。以下示例替换了上一步中显示的规则并添加计数器：

```
nft replace rule inet example_table example_chain handle 4 tcp dport 22 counter
accept
```

3.

显示计数器值：

```
nft list ruleset
table inet example_table {
 chain example_chain {
 type filter hook input priority filter; policy accept;
 tcp dport ssh counter packets 6872 bytes 105448565 accept
 }
}
```

#### 42.9.3. 监控与现有规则匹配的数据包

`nftables` 中的追踪功能与 `nft monitor` 命令相结合，使管理员能够显示与某一规则匹配的数据包。您可以为规则启用追踪，用它来监控匹配此规则的数据包。

##### 先决条件

- 您要添加计数器的规则已存在。

##### 流程

1.

在链中显示规则及其句柄：

```
nft --handle list chain inet example_table example_chain
table inet example_table {
 chain example_chain { # handle 1
 type filter hook input priority filter; policy accept;
 tcp dport ssh accept # handle 4
 }
}
```

2.

通过使用 `meta nftrace set 1` 参数替换规则来添加追踪功能。以下示例替换了上一步中显示的规则并启用追踪：

```
nft replace rule inet example_table example_chain handle 4 tcp dport 22 meta nftrace
set 1 accept
```

3.

使用 `nft monitor` 命令来显示追踪。以下示例过滤了命令的输出，来只显示包含 `inet example_table example_chain` 的条目：

```
nft monitor | grep "inet example_table example_chain"
trace id 3c5eb15e inet example_table example_chain packet: iif "enp1s0" ether saddr
52:54:00:17:ff:e4 ether daddr 52:54:00:72:2f:6e ip saddr 192.0.2.1 ip daddr 192.0.2.2 ip
dscp cs0 ip ecn not-ect ip ttl 64 ip id 49710 ip protocol tcp ip length 60 tcp sport 56728
tcp dport ssh tcp flags == syn tcp window 64240
trace id 3c5eb15e inet example_table example_chain rule tcp dport ssh nftrace set 1
accept (verdict accept)
...
...
```



警告

根据启用了追踪的规则数量以及匹配流量的数量，`nft monitor` 命令可以显示很多输出。使用 `grep` 或其他工具来过滤输出。

## 42.10. 备份和恢复 NFTABLES 规则集

您可以将 `nftables` 规则备份到文件，并稍后恢复它们。此外，管理员可以使用带有规则的文件，来将规则传给其他服务器。

### 42.10.1. 将 `nftables` 规则集备份到文件

您可以使用 `nft` 工具将 `nftables` 规则集备份到文件。

#### 流程

- 

要备份 `nftables` 规则：

- 以 `nft list ruleset` 格式生成的格式：

```
nft list ruleset > file.nft
```

- JSON 格式：

```
nft -j list ruleset > file.json
```

#### 42.10.2. 从文件中恢复 nftables 规则集

您可以从文件恢复 `nftables` 规则集。

##### 流程

- 要恢复 `nftables` 规则：
  - 如果要恢复的文件是 `nft list ruleset` 生成的格式，或直接包含 `nft` 命令：

```
nft -f file.nft
```

- 如果要恢复的文件采用 JSON 格式：

```
nft -j -f file.json
```

#### 42.11. 其它资源

- 在 Red Hat Enterprise Linux 8 中使用 `nftables`
- `iptables` 之后是什么？当然，它的继任者是：`nftables`
- FirewallD：未来是 `nftables`

## 第 43 章 使用 XDP-FILTER 进行高性能流量过滤以防止 DDOS 攻击

与 nftables 等数据包过滤器相比，Express Data Path(XDP)在网络接口处处理和丢弃网络数据包。因此，XDP 在到达防火墙或其他应用程序前决定了软件包的下一步。因此，XDP 过滤器需要较少的资源，处理网络数据包的速度要比传统数据包过滤器快得多，从而防止分布式拒绝服务(DDoS)攻击。例如，在测试过程中，红帽在单核上每秒丢弃了 2,600万个网络数据包，这比同一硬件上的 nftables 的丢弃率要高得多。

**xDP-filter** 工具使用 XDP 允许或丢弃传入的网络数据包。您可以创建规则来过滤与特定对象或特定命令的流量：

- IP 地址
- MAC 地址
- 端口

请注意，即使 xdp-filter 具有非常高的数据包处理率，但它不具有与 nftables 相同的功能。将 xdp-filter 视为一个概念性工具，来演示使用 XDP 的数据包过滤。另外，您可以使用工具代码来更好地了解如何编写您自己的 XDP 应用程序。



### 重要

在 AMD 和 Intel 64 位以外的构架上，xdp-filter 工具仅作为技术预览提供。红帽产品服务级别协议（SLA）不支持技术预览功能，且其功能可能并不完善，因此红帽不建议在生产环境中使用它们。这些预览可让用户早期访问将来的产品功能，让用户在开发过程中测试并提供反馈意见。

如需有关 [技术预览功能支持范围](#) 的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

### 43.1. 丢弃匹配 XDP-FILTER 规则的网络数据包

您可以使用 xdp-filter 丢弃网络数据包：

- 到特定目的地端口
- 从一个指定的 IP 地址
- 从一个指定的 MAC 地址

**xdp-filter** 的 **allow** 策略定义所有流量都被允许，过滤器只丢弃与特定规则匹配的网络数据包。例如，如果您知道要丢弃的数据包的源 IP 地址，请使用这个方法。

### 前提条件

- **xdp-tools** 软件包已安装。
- 支持 XDP 程序的网络驱动程序。

### 流程

1. 加载 **xdp-filter** 来处理特定接口上传入的数据包，如 **enp1s0**：

```
xdp-filter load enp1s0
```

默认情况下，**xdp-filter** 使用 **allow** 策略，并且工具只丢弃与任何规则匹配的流量。

(可选) 使用 **-f feature** 选项仅启用特定功能，如 **tcp**、**ipv4** 或 **etherent**。仅加载所需的功能而不是全部功能，可以提高数据包处理的速度。要启用多个功能，使用逗号分隔它们。

如果该命令出错，则网络驱动程序不支持 XDP 程序。

2. 添加规则来丢弃与它们匹配的数据包。例如：
  - 要将传入数据包放到端口 22，请输入：

```
xdp-filter port 22
```

这个命令添加一个匹配 TCP 和 UDP 流量的规则。要只匹配特定的协议，请使用 **-p protocol** 选项。

- 要丢弃来自 192.0.2.1 的传入数据包，请输入：

```
xdp-filter ip 192.0.2.1 -m src
```

请注意，**xdp-filter** 不支持 IP 范围。

- 要丢弃来自 MAC 地址 00:53:00:AA:07:BE 的传入数据包，请输入：

```
xdp-filter ether 00:53:00:AA:07:BE -m src
```

## 验证

- 使用以下命令显示丢弃和允许的数据包统计信息：

```
xdp-filter status
```

## 其他资源

- 您系统上 **xdp-filter (8)** 手册页
- 如果您是开发人员，并对 **xdp-filter** 代码感兴趣，请从红帽客户门户网站下载并安装相应的源 RPM (SRPM)。

### 43.2. 丢弃所有与 XDP-FILTER 规则匹配的网络数据包

您可以使用 **xdp-filter** 来只允许网络数据包：

- 来自和到一个特定目的地端口
- 来自和到一个特定 IP 地址
- 来自和到特定的 MAC 地址

要做到这一点，请使用 `xdf-filter` 的 `deny` 策略，其定义该过滤器会丢弃所有的网络数据包，与特定规则匹配的除外。例如，如果您不知道要丢弃的数据包的源 IP 地址，请使用这个方法。



### 警告

如果您在接口上加载 `xdf-filter` 时将默认策略设为 `deny`，则内核会立即丢弃来自这个接口的所有数据包，直到您创建允许某些流量的规则。要避免从系统中锁定，在本地输入命令或者通过不同的网络接口连接到主机。

### 前提条件

- `xdf-tools` 软件包已安装。
- 您登录到本地主机，或使用您不计划过滤流量的网络接口。
- 支持 XDP 程序的网络驱动程序。

### 流程

1. 加载 `xdf-filter` 来处理特定接口上的数据包，如 `enp1s0`：

```
xdf-filter load enp1s0 -p deny
```

(可选) 使用 `-f feature` 选项仅启用特定功能，如 `tcp`、`ipv4` 或 `etherent`。仅加载所需的功

能而不是全部功能，可以提高数据包处理的速度。要启用多个功能，使用逗号分隔它们。

如果该命令出错，则网络驱动程序不支持 XDP 程序。

2.

添加规则以允许匹配它们的数据包。例如：

- 要允许数据包发送端口 22，请输入：

```
xdp-filter port 22
```

这个命令添加一个匹配 TCP 和 UDP 流量的规则。要只匹配特定的协议，请将 `-p protocol` 选项传给命令。

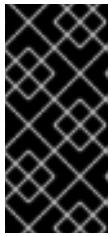
- 要允许数据包发送到 192.0.2.1，请输入：

```
xdp-filter ip 192.0.2.1
```

请注意，`xdp-filter` 不支持 IP 范围。

- 要允许数据包发送到 MAC 地址 00:53:00:AA:07:BE，请输入：

```
xdp-filter ether 00:53:00:AA:07:BE
```

 重要

`xdp-filter` 工具不支持有状态数据包检查。这要求您不使用 `-m mode` 选项设置模式，或者您添加显式规则来允许机器接收传入流量来作为对传出流量的答复。

验证

- 使用以下命令显示丢弃和允许的数据包统计信息：

```
xdp-filter status
```

## 其他资源

- 您系统上 **xdp-filter (8)** 手册页
- 如果您是开发人员，并且您对 **xdp-filter** 的代码感兴趣，请从红帽客户门户网站下载并安装相应的源 RPM (SRPM)。

## 第 44 章 捕获网络数据包

要调试网络问题和通讯，您可以捕获网络数据包。以下部分提供有关捕获网络数据包的步骤和附加信息。

### 44.1. 使用 XDPDUMP 捕获包括 XDP 程序丢弃的数据包在内的网络数据包

`xpdump` 工具捕获网络数据包。与 `tcpdump` 工具不同，`xpdump` 为此使用扩展的 Berkeley 数据包过滤(eBPF)程序。这也可使 `xpdump` 能够捕获快速数据路径(XDP)程序丢弃的数据包。用户空间工具（如 `tcpdump`）无法捕获这些被丢弃的数据包，以及 XDP 程序修改的原始数据包。

您可以使用 `xpdump` 来调试已附加到接口上的 XDP 程序。因此，实用程序可以在 XDP 程序启动和完成后捕获数据包。在后一种情况下，`xpdump` 也捕获 XDP 操作。默认情况下，`xpdump` 会在 XDP 程序的入口处捕获传入的数据包。



#### 重要

在 AMD 和 Intel 64 位以外的其他构架上，`xpdump` 工具仅作为技术预览提供。红帽产品服务级别协议（SLA）不支持技术预览功能，且其功能可能并不完善，因此红帽不建议在生产环境中使用它们。这些预览可让用户早期访问将来的产品功能，让用户在开发过程中测试并提供反馈意见。

如需有关 [技术预览功能支持范围](#) 的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

请注意，`xpdump` 没有数据包过滤或解码功能。但是，您可以将它与 `tcpdump` 结合使用来解码数据包。

#### 先决条件

- 支持 XDP 程序的网络驱动程序。
- XDP 程序被加载到 `enp1s0` 接口。如果没有程序载入，`xpdump` 会以与 `tcpdump` 类似的方式捕获数据包，以便向后兼容。

## 流程

1.

要捕获 `enp1s0` 接口上的数据包，并将它们写入到 `/root/capture.pcap` 文件，请输入：

```
xdpdump -i enp1s0 -w /root/capture.pcap
```

2.

要停止捕获数据包，请按 `Ctrl+C`。

## 其他资源

- 您系统上的 `xpdump (8)` 手册页
- 如果您是开发人员，并且您对 `xpdump` 的源代码感兴趣，请从红帽客户门户网站下载并安装相应的源 `RPM(SRPM)`。

### 44.2. 其他资源

- [如何使用 `tcpdump` 捕获网络数据包？（红帽知识库）](#)

## 第 45 章 了解 RHEL 8 中的 EBPF 网络功能

扩展的 Berkeley Packet 过滤器 (eBPF) 是一个内核中的虚拟机，允许在内核空间中执行代码。此代码运行在一个受限的沙箱环境中，仅可访问有限功能集。

在网络中，您可以使用 eBPF 来补充或替换内核数据包处理。根据您使用的 hook，eBPF 程序有：

- 对元数据的读和写的访问权限
- 可以查找套接字和路由
- 可以设置套接字选项
- 可以重定向数据包

### 45.1. RHEL 8 中网络 EBPF 功能概述

您可以将扩展的 Berkeley 数据包过滤器(eBPF)网络程序附加到 RHEL 中的以下钩子：

- **Express Data Path(XDP)**：在内核网络堆栈处理它们之前，对接收的数据包提供早期的访问权限。
- 带有直接动作标志的 tc eBPF 分类器：对入口和出口提供强大的数据包处理。
- **控制组版本 2(cgroup v2)**：在控制组中，对程序所执行的基于套接字的操作启用过滤和覆盖。
- **套接字过滤**：启用对从套接字接收的数据包进行过滤。这个功能也可用于经典 Berkeley Packet Filter (cBPF)，但已扩展为支持 eBPF 程序。

- 流解析器：启用将流分成单独的消息、过滤并将其重定向到套接字。
- **SO\_REUSEPORT 套接字选择**：对来自 `reuseport` 套接字组的接收套接字提供可编程选择。
- 流程分析器：在某些情况下，启用覆盖内核解析数据包头的方式。
- **TCP 拥塞控制回调**：启用实现一个自定义 TCP 拥塞控制算法。
- 带有封装的路由：允许创建自定义隧道封装。

请注意，红帽并不支持 RHEL 中的所有 eBPF 功能，如下所述。如需了解更多与每个 hook 相关的信息，请参阅 [RHEL 8 发行注记](#) 和以下概述。

## XDP

您可以将 `BPF_PROG_TYPE_XDP` 类型的程序附加到网络接口。然后，在内核网络堆栈开始处理之前，内核会在接收的数据包上执行该程序。在某些情况下，这允许快速数据包转发，如快速数据包丢弃以防止分布式拒绝服务(DDoS)攻击，以及负载均衡场景的快速数据包重定向。

您还可以使用 XDP 进行不同类型的数据包监控和抽样。内核允许 XDP 程序修改数据包，并将其传送到内核网络堆栈进行进一步处理。

以下的 XDP 模式可用：

- **原生（驱动程序）XDP**：内核在数据包接收过程从最早可能的点执行程序。目前，内核无法解析数据包，因此无法使用内核提供的元数据。这个模式要求网络接口驱动程序支持 XDP，但并非所有驱动程序都支持这种原生模式。
- **通用 XDP**：内核网络栈在进程早期执行 XDP 程序。此时内核数据结构已被分配，数据包已被预先处理。如果数据包被丢弃或重定向，与原生模式相比，这需要大量开销。但是，通用模式不需要支持网络接口驱动，它可适用于所有网络接口。
-

**Offloaded XDP**：内核在网络接口而不是主机 CPU 上执行 XDP 程序。请注意，这需要特定的硬件，这个模式中只有某些 eBPF 功能可用。

在 RHEL 上，使用 libxdp 库加载所有 XDP 程序。这个程序库启用系统控制的 XDP 使用。



#### 注意

目前，XDP 程序有一些系统配置限制。例如：您必须禁用接收接口中某些硬件卸载功能。另外，并非所有功能都可用于支持原生模式的所有驱动程序。

在 RHEL 8.7 中，红帽仅在满足以下条件时才支持 XDP 功能：

- 您可以在 AMD 或者 Intel 64 位构架中载入 XDP 程序。
- 您可以使用 libxdp 库将程序加载到内核中。
- XDP 程序不使用 XDP 硬件卸载。

另外，红帽还提供以下使用 XDP 功能作为不受支持的技术预览：

- 在 AMD 和 Intel 64 位以外的构架中载入 XDP 程序。请注意，libxdp 库不适用于 AMD 和 Intel 64 位的构架。
- XDP 硬件卸载。

## AF\_XDP

使用过滤并将数据包重定向到给定的 AF\_XDP 套接字的 XDP 程序，您可以使用 AF\_XDP 协议系列中的一个或多个套接字来快速将数据包从内核复制到用户空间。

在 RHEL 8.7 中，红帽将此功能作为不受支持的技术预览提供。

## 流量控制

流量控制(tc)子系统提供以下 eBPF 程序类型：

- **BPF\_PROG\_TYPE\_SCHED\_CLS**
- **BPF\_PROG\_TYPE\_SCHED\_ACT**

这些类型允许您在 eBPF 中编写自定义的 tc 分类器和 tc 操作。与 tc 生态系统的各个部分一起，这为强大的数据包处理提供了能力，是一些容器编排解决方案的核心部分。

在大多数情况下，只有类符被使用，与 direct-action 标记一样，eBPF 分类器可以直接从同一 eBPF 程序执行操作。`clsact` 排队规程(qdisc)被设计为在入口端启用此功能。

请注意，使用流解析器 eBPF 程序可能会影响其他 qdiscs 和 tc 分类器的操作，如 flower。

RHEL 8.2 及更新的版本完全支持 tc 特性的 eBPF。

## 套接字过滤器

一些实用程序会使用或在过去使用了 classic Berkeley Packet Filter (cBPF) 过滤套接字上接收到的数据包。例如，tcpdump 工具允许用户指定表达式，tcpdump 然后将它们转换为 cBPF 码。

作为 cBPF 的替代方案，内核允许 **BPF\_PROG\_TYPE\_SOCKET\_FILTER** 类型的 eBPF 程序实现相同的目的。

在 RHEL 8.7 中，红帽将此功能作为不受支持的技术预览提供。

## 控制组群

在 RHEL 中，您可以使用多种 eBPF 程序，供您附加到 cgroup。当给定 cgroup 中的某个程序执行某个操作时，内核会执行这些程序。请注意，您只能使用 cgroups 版本 2。

RHEL 中提供以下与网络相关的 cgroup eBPF 程序：

- **BPF\_PROG\_TYPE SOCK\_OPS**：内核对各种 TCP 事件调用该程序。程序可以调整内核 TCP 堆栈的行为，包括自定义 TCP 头选项等。
- **BPF\_PROG\_TYPE\_CGROUP\_SOCK\_ADDR**：在 connect、bind、sendto、recvmsg、getpeername 和 getsockname 操作过程中，内核调用该程序。该程序允许更改 IP 地址和端口。当您在 eBPF 中实现基于套接字的网络地址转换(NAT)时，这很有用。
- **BPF\_PROG\_TYPE\_CGROUP\_SOCKOPT**：在 setsockopt 和 getsockopt 过程中，内核调用该程序，并允许更改选项。
- **BPF\_PROG\_TYPE\_CGROUP\_SOCK**：在套接字创建、套接字释放和绑定到地址的过程中，内核调用该程序。您可以使用这些程序来允许或拒绝操作，或者只检查套接字创建统计信息。
- **BPF\_PROG\_TYPE\_CGROUP\_SKB**：该程序在入口和出口处过滤单个数据包，并可以接受或拒绝数据包。
- **BPF\_CGROUP\_INET4\_GETPEERNAME、BPF\_CGROUP\_INET6\_GETPEERNAME、BPF\_CGROUP\_INET4\_GETSOCKNAME 和 BPF\_CGROUP\_INET6\_GETSOCKNAME**：使用这些程序，您可以覆盖 getpeername 和 getsockname 系统调用的结果。当您在 eBPF 中实现基于套接字的网络地址转换(NAT)时，这很有用。

在 RHEL 8.7 中，红帽将此功能作为不受支持的技术预览提供。

### 流解析器（Stream Parser）

流解析器对添加到特殊 eBPF 映射中的一组套接字进行操作。然后 eBPF 程序处理内核在那些套接字上接收或发送的数据包。

RHEL 中提供了以下流解析程序 eBPF 程序：

- **BPF\_PROG\_TYPE\_SK\_SKB** : eBPF 程序将来自套接字的数据包解析为单独的消息，并指示内核丢弃这些消息或将其发送给组中的另一个套接字。
- **BPF\_PROG\_TYPE\_SK\_MSG** : 此程序过滤出口消息。eBPF 程序将数据包解析到单个信息中，并批准或拒绝它们。

在 RHEL 8.7 中，红帽将此功能作为不受支持的技术预览提供。

### SO\_REUSEPORT 套接字选择

使用这个套接字选项，您可以绑定多个套接字到相同的 IP 地址和端口。如果没有 eBPF，内核会根据连接散列选择接收套接字。有了 **BPF\_PROG\_TYPE\_SK\_REUSEPORT** 程序，接收套接字的选择是完全可编程的。

在 RHEL 8.7 中，红帽将此功能作为不受支持的技术预览提供。

### dissector 流程

当内核需要处理数据包头，而不需要查看全部协议解码时，会对它们进行剖析。例如，这会在 tc 子系统、多路径路由、绑定或者计算数据包哈希时发生。在这种情况下，内核解析数据包的标头，并使用数据包标头中的信息填充内部结构。您可以使用 **BPF\_PROG\_TYPE\_FLOW\_DISSECTOR** 程序替换此内部解析。请注意，您只能在 RHEL 的 eBPF 的 IPv4 和 IPv6 上分离 TCP 和 UDP。

在 RHEL 8.7 中，红帽将此功能作为不受支持的技术预览提供。

### TCP 阻塞控制

您可以使用一组实现 `struct tcp_congestion_ops` 回调的 **BPF\_PROG\_TYPE\_STRUCT\_OPS** 程序来编写一个自定义的 TCP 阻塞控制算法。以这种方式实现的算法和内置的内核算法一起可提供给系统使用。

在 RHEL 8.7 中，红帽将此功能作为不受支持的技术预览提供。

### 带有封装的路由

您可以将以下 eBPF 程序类型之一附加到路由表中作为隧道封装属性的路由：

- **BPF\_PROG\_TYPE\_LWT\_IN**
- **BPF\_PROG\_TYPE\_LWT\_OUT**
- **BPF\_PROG\_TYPE\_LWT\_XMIT**

这样的 eBPF 程序的功能仅限于特定的隧道配置，它不允许创建通用封装或封装解决方案。

在 RHEL 8.7 中，红帽将此功能作为不受支持的技术预览提供。

## 套接字查找

要绕过 bind 系统调用的限制，请使用 BPF\_PROG\_TYPE\_SK\_LOOKUP 类型的 eBPF 程序。此类程序可以为新传入的 TCP 连接选择侦听套接字，或为 UDP 数据包选择一个未连接的套接字。

在 RHEL 8.7 中，红帽将此功能作为不受支持的技术预览提供。

## 45.2. 按网卡的 RHEL 8 中的 XDP 功能概述

以下是启用了 XDP 的网卡和您可以使用的 XDP 特性的概述：

网卡	驱动	基本的	重定向	目标	HW 卸载	零复制
Amazon 弹性网络适配卡	<b>ena</b>	是	是	是 [a]	否	否
Broadcom NetXtreme-C/E 10/25/40/50 千兆以太网	<b>bnxt_en</b>	是	是	是 [a]	否	否
Cavium Thunder 虚拟功能	<b>nicvf</b>	是	否	否	否	否
Google Virtual NIC (gVNIC) 支持	<b>gve</b>	是	是	是	否	是
Intel® 10GbE PCI Express 虚拟功能以太网	<b>ixgbbevf</b>	是	否	否	否	否

网卡	驱动	基本的	重定向	目标	HW 卸载	零复制
Intel® 10GbE PCI Express 适配卡	<b>ixgbe</b>	是	是	是 [a]	否	是
Intel® 以太网连接 E800 系列	<b>ice</b>	是	是	是 [a]	否	是
Intel® Ethernet Controller I225-LM/I225-V 系列	<b>igc</b>	是	是	是	否	是
Intel® 以太网控制器 XL710 系列	<b>i40e</b>	是	是	是 [a] [b]	否	是
Intel® PCI Express 千兆适配卡	<b>igb</b>	是	是	是 [a]	否	否
Mellanox 第 5 代网络适配卡 (ConnectX 系列)	<b>mlx5_core</b>	是	是	是 [b]	否	是
Mellanox Technologies 1/10/40Gbit 以太网	<b>mlx4_en</b>	是	是	否	否	否
Microsoft Azure Network Adapter	<b>mana</b>	是	是	是	否	否
Microsoft Hyper-V 虚拟网络	<b>hv_netvsc</b>	是	是	是	否	否
Netronome® NFP4000/NFP6000 NIC	<b>nfp</b>	是	否	否	是	否
QEMU Virtio 网络	<b>virtio_net</b>	是	是	是 [a]	否	否
QLogic QED 25/40/100Gb 以太网 NIC	<b>qede</b>	是	是	是	否	否
Solarflare SFC9000/SFC9100/EF100-系列	<b>sfc</b>	是	是	是 [b]	否	否
通用 TUN/TAP 设备	<b>tun</b>	是	是	是	否	否
虚拟以太网对设备	<b>veth</b>	是	是	是	否	否

[a] 只有在接口上加载 XDP 程序时。

[b] 需要分配几个大于或等于最大 CPU 索引的 XDP TX 队列。

图例：

- 基本的：支持基本的返回代码：**DROP**、**PASS**、**ABORTED** 和 **TX**。
- 重定向：支持 **REDIRECT** 返回码。
- 目标：可以是 **REDIRECT** 返回码的目标。
- HW 卸载：支持 **XDP** 硬件卸载。
- 零-复制：支持 **AF\_XDP** 协议系列的零复制模式。

## 第 46 章 使用 BPF 编译器集合进行网络追踪

**BPF Compiler Collection (BCC)** 是一个库，可帮助创建扩展的 Berkeley Packet Filter (eBPF) 程序。eBPF 程序的主要工具是分析操作系统性能和网络性能，而不会遇到开销或安全问题。

BCC 不再需要用户了解 eBPF 的技术详情，并提供了许多开箱即用的起点，如带有预先创建的 eBPF 程序的 **bcc-tools** 软件包。



### 注意

eBPF 程序在事件中触发，如磁盘 I/O、TCP 连接以及进程创建。程序不太可能导致内核崩溃、循环或者变得无响应，因为它们在内核的安全性虚拟机中运行。

#### 46.1. 安装 BCC-TOOLS 软件包

安装 **bcc-tools** 软件包，该软件包还会将 **BPF Compiler Collection (BCC)** 库作为依赖项安装。

### 流程

- 

安装 **bcc-tools**。

```
yum install bcc-tools
```

BCC 工具安装在 `/usr/share/bcc/tools/` 目录中。

### 验证

- 

检查安装的工具：

```
ls -l /usr/share/bcc/tools/
...
-rwxr-xr-x. 1 root root 4198 Dec 14 17:53 dcsnoop
-rwxr-xr-x. 1 root root 3931 Dec 14 17:53 dcstat
-rwxr-xr-x. 1 root root 20040 Dec 14 17:53 deadlock_detector
-rw-r--r--. 1 root root 7105 Dec 14 17:53 deadlock_detector.c
drwxr-xr-x. 3 root root 8192 Mar 11 10:28 doc
-rwxr-xr-x. 1 root root 7588 Dec 14 17:53 execsnoop
```

```
-rwxr-xr-x. 1 root root 6373 Dec 14 17:53 ext4dist
-rwxr-xr-x. 1 root root 10401 Dec 14 17:53 ext4slower
...
```

列表中的 doc 目录提供每个工具的文档。

## 46.2. 显示添加到内核的接受队列中的 TCP 连接

内核在 TCP 3 向握手中接收 ACK 数据包后，内核会将来自 SYN 队列的连接移到 accept 队列，直到连接的状态变为 ESTABLISHED。因此，只有成功的 TCP 连接才能在此队列中看到。

`tcpaccept` 工具使用 eBPF 特性显示内核添加到 accept 队列的所有连接。该工具是轻量级的，因为它跟踪内核的 `accept()` 函数，而不是捕获和过滤数据包。例如，使用 `tcpaccept` 进行常规故障排除，来显示服务器已接受的新连接。

### 流程

1. 输入以下命令来启动对内核 accept 队列的追踪：

```
/usr/share/bcc/tools/tcpaccept
PID COMM IP RADDR RPORT LADDR LPORT
843 sshd 4 192.0.2.17 50598 192.0.2.1 22
1107 ns-slapd 4 198.51.100.6 38772 192.0.2.1 389
1107 ns-slapd 4 203.0.113.85 38774 192.0.2.1 389
...
```

每次内核接受一个连接时，`tcpaccept` 都会显示连接的详情。

2. 按 **Ctrl+C** 停止追踪过程。

### 其他资源

- 您系统上的 `tcpaccept (8)` 手册页
- `/usr/share/bcc/tools/doc/tcpaccept_example.txt` 文件

### 46.3. 追踪出去的 TCP 连接尝试

**tcpconnect** 工具使用 eBPF 特性来跟踪出去的 TCP 连接尝试。该工具的输出还包括失败的连接。

**tcpconnect** 工具是轻量级的，例如，因为它跟踪内核的 `connect()` 函数，而不是捕获和过滤数据包。

#### 流程

1.

输入以下命令启动显示所有传出连接的追踪过程：

```
/usr/share/bcc/tools/tcpconnect
PID COMM IP SADDR DADDR DPORT
31346 curl 4 192.0.2.1 198.51.100.16 80
31348 telnet 4 192.0.2.1 203.0.113.231 23
31361 isc-worker00 4 192.0.2.1 192.0.2.254 53
...
```

每次内核处理一个出去的连接时，**tcpconnect** 都会显示连接的详情。

2.

按 **Ctrl+C** 停止追踪过程。

#### 其他资源

- 您系统上的 **tcpconnect (8)** 手册页
- **/usr/share/bcc/tools/doc/tcpconnect\_example.txt** 文件

### 46.4. 测量出站 TCP 连接的延迟

TCP 连接延迟是建立连接所需的时间。这通常涉及内核 TCP/IP 处理和网络往返时间，而不是应用程序运行时。

**tcpconnlat** 工具使用 eBPF 特性来测量发送 SYN 数据包和接收响应数据包之间的时间。

## 流程

1.

开始测量出站连接的延迟：

```
/usr/share/bcc/tools/tcpconnlat
PID COMM IP SADDR DADDR DPORT LAT(ms)
32151 isc-worker00 4 192.0.2.1 192.0.2.254 53 0.60
32155 ssh 4 192.0.2.1 203.0.113.190 22 26.34
32319 curl 4 192.0.2.1 198.51.100.59 443 188.96
...
```

每次内核处理一个出去的连接时，`tcpconnlat` 都会在内核接收响应数据包后显示连接的详细信息。

2.

按 **Ctrl+C** 停止追踪过程。

## 其他资源

- 您系统上的 `tcpconnlat (8)` 手册页
- `/usr/share/bcc/tools/doc/tcpconnlat_example.txt` 文件

## 46.5. 显示被内核丢弃的 TCP 数据包和片段详情

`tcpdrop` 工具使管理员能够显示内核所丢弃的 TCP 数据包和段的详情。使用这个实用程序调试丢弃数据包的高速率，以便远程系统发送基于计时器的重新传输。释放数据包和片段的高速率可能会影响服务器的性能。

`tcpdrop` 工具使用 eBPF 特性，而不是捕获和过滤资源密集型的数据包，来直接从内核检索信息。

## 流程

1.

输入以下命令来显示丢弃 TCP 数据包和片段详情：

```
/usr/share/bcc/tools/tcpdrop
TIME PID IP SADDR:SPORT > DADDR:DPORT STATE (FLAGS)
13:28:39 32253 4 192.0.2.85:51616 > 192.0.2.1:22 CLOSE_WAIT (FIN|ACK)
```

```
b'tcp_drop+0x1'
b'tcp_data_queue+0x2b9'
...
13:28:39.1 4 192.0.2.85:51616 > 192.0.2.1:22 CLOSE (ACK)
b'tcp_drop+0x1'
b'tcp_rcv_state_process+0xe2'
...
```

每次内核丢弃 TCP 数据包和段时，`tcpdrop` 都会显示连接的详情，包括导致软件包丢弃的内核堆栈跟踪。

- 2.
- 按 **Ctrl+C** 停止追踪过程。

## 其他资源

- 您系统上的 `tcpdrop (8)` 手册页
- `/usr/share/bcc/tools/doc/tcpdrop_example.txt` 文件

## 46.6. 追踪 TCP 会话

`tcplife` 工具使用 eBPF 跟踪打开和关闭的 TCP 会话，并打印一行输出来总结每一个会话。管理员可以使用 `tcplife` 来识别连接和传输的流量数。

例如，您可以显示到端口 22 (SSH) 的连接来检索以下信息：

- 本地进程 ID (PID)
- 本地进程名称
- 本地 IP 地址和端口号

- 远程 IP 地址和端口号
- 接收和传输的流量的数量（以 KB 为单位）。
- 连接处于活跃状态的时间（毫秒）

## 流程

1. 输入以下命令来开始追踪到本地端口 22 的连接：

```
/usr/share/bcc/tools/tcpdump -L 22
PID COMM LADDR LPORT RADDR RPORT TX_KB RX_KB MS
19392 sshd 192.0.2.1 22 192.0.2.17 43892 53 52 6681.95
19431 sshd 192.0.2.1 22 192.0.2.245 43902 81 249381 7585.09
19487 sshd 192.0.2.1 22 192.0.2.121 43970 6998 7 16740.35
...
```

每次关闭连接时，`tcpdump` 都会显示连接的详情。

2. 按 `Ctrl+C` 停止追踪过程。

## 其他资源

- 您系统上的 `tcpdump (8)` 手册页
- `/usr/share/bcc/tools/doc/tcpdump_example.txt` 文件

## 46.7. 追踪 TCP 重新传输

`tcpretrans` 工具显示有关 TCP 重新传输的详细信息，如本地和远程的 IP 地址和端口号，以及重新传输时 TCP 的状态。

该工具使用 eBPF 功能，因此开销非常低。

## 流程

1.

使用以下命令来显示 TCP 重新传输详情：

```
/usr/share/bcc/tools/tcpretrans
TIME PID IP LADDR:LPORT T> RADDR:RPORT STATE
00:23:02 0 4 192.0.2.1:22 R> 198.51.100.0:26788 ESTABLISHED
00:23:02 0 4 192.0.2.1:22 R> 198.51.100.0:26788 ESTABLISHED
00:45:43 0 4 192.0.2.1:22 R> 198.51.100.0:17634 ESTABLISHED
...
...
```

每次内核调用 TCP 重新传输函数时，`tcpretrans` 都会显示连接的详情。

2.

按 **Ctrl+C** 停止追踪过程。

## 其他资源

- 您系统上的 `tcpretrans (8)` 手册页
- `/usr/share/bcc/tools/doc/tcpretrans_example.txt` 文件

## 46.8. 显示 TCP 状态更改信息

在 TCP 会话中，TCP 状态会改变。`tcpstates` 工具使用 eBPF 功能跟踪这些状态变化，并打印包括每个状态持续时间的详细信息。例如，使用 `tcpstates` 来确定连接是否在初始化状态中花费了太多时间。

## 流程

1.

使用以下命令开始追踪 TCP 状态变化：

```
/usr/share/bcc/tools/tcpstates
SKADDR C-PID C-COMM LADDR LPORT RADDR RPORT OLDSTATE ->
NEWSTATE MS
ffff9cd377b3af80 0 swapper/1 0.0.0.0 22 0.0.0.0 0 LISTEN -> SYN_RECV
0.000
ffff9cd377b3af80 0 swapper/1 192.0.2.1 22 192.0.2.45 53152 SYN_RECV ->
ESTABLISHED 0.067
ffff9cd377b3af80 818 sssd_nss 192.0.2.1 22 192.0.2.45 53152 ESTABLISHED ->
CLOSE_WAIT 65636.773
```

```
ffff9cd377b3af80 1432 sshd 192.0.2.1 22 192.0.2.45 53152 CLOSE_WAIT ->
LAST_ACK 24.409
ffff9cd377b3af80 1267 pulseaudio 192.0.2.1 22 192.0.2.45 53152 LAST_ACK ->
CLOSE 0.376
...
```

每次连接改变其状态时，`tcpstates` 都会显示一个新行，其中包含更新的连接详情。

如果多个连接同时改变了其状态，请使用第一列中的套接字地址(SKADDR)来确定哪些条目属于同一个连接。

2. 按 **Ctrl+C** 停止追踪过程。

## 其他资源

- 您系统上的 `tcpstates (8)` 手册页
- `/usr/share/bcc/tools/doc/tcpstates_example.txt` 文件

### 46.9. 聚合发送到特定子网的 TCP 流量

`tcpsubnet` 工具汇总并合计了本地主机发往子网的 IPv4 TCP 流量，并按固定间隔显示输出。该工具使用 eBPF 功能来收集并总结数据，以减少开销。

默认情况下，`tcpsubnet` 为以下子网汇总流量：

- **127.0.0.1/32**
- **10.0.0.0/8**
- **172.16.0.0/12**

- 192.0.2.0/24/16
- 0.0.0.0/0

请注意，最后一个子网(0.0.0.0/0)是一个全包括选项。tcpsubnet 工具计算与这个全包括条目中前四个不同的子网的所有流量。

按照以下流程计算 192.0.2.0/24 和 198.51.100.0/24 子网的流量。到其他子网的流量将在 0.0.0.0/0 全包括子网条目中跟踪。

## 流程

1.

开始监控发送到 192.0.2.0/24、198.51.100.0/24 和其他子网的流量数：

```
/usr/share/bcc/tools/tcpsubnet 192.0.2.0/24,198.51.100.0/24,0.0.0.0/0
Tracing... Output every 1 secs. Hit Ctrl-C to end
[02/21/20 10:04:50]
192.0.2.0/24 856
198.51.100.0/24 7467
[02/21/20 10:04:51]
192.0.2.0/24 1200
198.51.100.0/24 8763
0.0.0.0/0 673
...
```

这个命令以字节为单位显示指定子网每秒一次的流量。

2.

按 Ctrl+C 停止追踪过程。

## 其他资源

- 您系统上的 **tcpsubnet (8)** 手册页
- **/usr/share/bcc/tools/doc/tcpsubnet.txt** 文件

## 46.10. 通过 IP 地址和端口显示网络吞吐量

**tcptop** 工具以 KB 为单位显示主机发送并接收的 TCP 流量。这个报告会自动刷新并只包含活跃的 TCP 连接。该工具使用 eBPF 功能，因此开销非常低。

### 流程

- 1.

要监控发送和接收的流量，请输入：

```
/usr/share/bcc/tools/tcptop
13:46:29 loadavg: 0.10 0.03 0.01 1/215 3875

PID COMM LADDR RADDR RX_KB TX_KB
3853 3853 192.0.2.1:22 192.0.2.165:41838 32 102626
1285 sshd 192.0.2.1:22 192.0.2.45:39240 0 0
...
```

命令的输出只包括活跃的 TCP 连接。如果本地或者远程系统关闭了连接，则该连接在输出中不再可见。

- 2.

按 **Ctrl+C** 停止追踪过程。

### 其他资源

- 您系统上的 **tcptop (8)** 手册页
- **/usr/share/bcc/tools/doc/tcptop.txt** 文件

## 46.11. 追踪已建立的 TCP 连接

**tcptracer** 工具跟踪连接、接受和关闭 TCP 连接的内核函数。该工具使用 eBPF 功能，因此开销非常低。

### 流程

- 1.

使用以下命令启动追踪过程：

```
/usr/share/bcc/tools/tcptracer
Tracing TCP established connections. Ctrl-C to end.
T PID COMM IP SADDR DADDR SPORT DPORt
A 1088 ns-slapd 4 192.0.2.153 192.0.2.1 0 65535
A 845 sshd 4 192.0.2.1 192.0.2.67 22 42302
X 4502 sshd 4 192.0.2.1 192.0.2.67 22 42302
...
```

每当内核连接、接受或关闭连接时，`tcptracer` 都会显示连接的详情。

2. 按 **Ctrl+C** 停止追踪过程。

## 其他资源

- 您系统上的 `tcptracer (8)` 手册页
- `/usr/share/bcc/tools/doc/tcptracer_example.txt` 文件

## 46.12. 追踪 IPV4 和 IPV6 倾听尝试

`solisten` 工具追踪所有 IPv4 和 IPv6 倾听尝试。它跟踪尝试，包括最终失败的或者不接受连接的倾听程序。当程序希望倾听 TCP 连接时，工具会跟踪内核调用的函数。

## 流程

1. 输入以下命令启动显示所有监听 TCP 尝试的追踪过程：

```
/usr/share/bcc/tools/solisten
PID COMM PROTO BACKLOG PORT ADDR
3643 nc TCPv4 1 4242 0.0.0.0
3659 nc TCPv6 1 4242 2001:db8:1::1
4221 redis-server TCPv6 128 6379 ::
4221 redis-server TCPv4 128 6379 0.0.0.0
....
```

2. 按 **Ctrl+C** 停止追踪过程。

## 其他资源

- 您系统上的 **solisten (9)** 手册页
- **/usr/share/bcc/tools/doc/solisten\_example.txt** 文件

### 46.13. 软中断的服务时间概述

**softirqs** 工具总结了服务软中断（soft IRQ）所花费的时间，并将这个时间显示为总计或直方图分布。这个工具使用 `irq:softirq_enter` 和 `irq:softirq_exit` 内核追踪点，是一个稳定的追踪机制。

#### 流程

1. 输入以下命令启动追踪 soft irq 事件时间：

```
/usr/share/bcc/tools/softirqs
Tracing soft irq event time... Hit Ctrl-C to end.
^C
SOFTIRQ TOTAL_usecs
tasklet 166
block 9152
net_rx 12829
rcu 53140
sched 182360
timer 306256
```

2. 按 **Ctrl+C** 停止追踪过程。

#### 其他资源

- 您系统上的 **softirqs (8)** 和 **mpstat (1)** 手册页
- **/usr/share/bcc/tools/doc/softirqs\_example.txt** 文件

### 46.14. 总结网络接口上的数据包大小和数量

**netqtop** 工具显示有关特定网络接口的每个网络队列上收到的(RX)和传输的(TX)数据包属性的统计信息。统计包括：

- 每秒字节数(BPS)
- 每秒数据包数(PPS)
- 平均数据包大小
- 总数据包数

要生成这些统计数据，`netqtop` 会跟踪执行传输的数据包 `net_dev_start_xmit` 以及接收的数据包 `netif_receive_skb` 的事件的内核功能。

## 流程

1. 显示 2 秒时间间隔的字节大小范围内的数据包数：

```
/usr/share/bcc/tools/netqtop -n enp1s0 -i 2

Fri Jan 31 18:08:55 2023
TX
QueueID avg_size [0, 64) [64, 512) [512, 2K) [2K, 16K) [16K, 64K)
0 0 0 0 0 0 0
Total 0 0 0 0 0 0

RX
QueueID avg_size [0, 64) [64, 512) [512, 2K) [2K, 16K) [16K, 64K)
0 38.0 1 0 0 0 0
Total 38.0 1 0 0 0 0

Fri Jan 31 18:08:57 2023
TX
QueueID avg_size [0, 64) [64, 512) [512, 2K) [2K, 16K) [16K, 64K)
0 0 0 0 0 0 0
Total 0 0 0 0 0 0

RX
QueueID avg_size [0, 64) [64, 512) [512, 2K) [2K, 16K) [16K, 64K)
0 38.0 1 0 0 0 0
Total 38.0 1 0 0 0 0

```

2.

按 **Ctrl+C** 来停止 **netqtop**。

## 其他资源

- 您系统上的 **netqtop (8)** 手册页
- **/usr/share/bcc/tools/doc/netqtop\_example.txt**

## 第 47 章 配置网络设备以接受来自所有 MAC 地址的流量

网络设备通常会拦截和读取编程的控制器接收的数据包。您可以在虚拟交换机或端口组层面上，将网络设备配置为接受来自所有 MAC 地址的流量。

您可以使用这个网络模式来：

- 诊断网络连接问题
- 出于安全原因监控网络活动
- 拦截传输中的私有数据或网络中的入侵

您可以为任何类型的网络设备启用此模式，除了 InfiniBand。

### 47.1. 临时配置设备以接受所有流量

您可以使用 ip 工具临时配置网络设备以接受所有流量，而不考虑 MAC 地址。

#### 流程

1.

可选：显示网络接口以标识您要接收所有流量的接口：

```
ip address show
1: enp1s0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel
state DOWN group default qlen 1000
 link/ether 98:fa:9b:a4:34:09 brd ff:ff:ff:ff:ff:ff
 ...

```

2.

修改设备以启用或禁用此属性：

- 为 enp1s0 启用 accept-all-mac-addresses 模式：

```
ip link set enp1s0 promisc on
```

- 为 `enp1s0` 禁用 `accept-all-mac-addresses` 模式：

```
ip link set enp1s0 promisc off
```

## 验证

- 验证 `accept-all-mac-addresses` 模式是否已启用：

```
ip link show enp1s0
1: enp1s0: <NO-CARRIER,BROADCAST,MULTICAST,PROMISC,UP> mtu 1500 qdisc fq_codel
state DOWN mode DEFAULT group default qlen 1000
link/ether 98:fa:9b:a4:34:09 brd ff:ff:ff:ff:ff:ff
```

设备描述中的 `PROMISC` 标志表示启用了该模式。

## 47.2. 使用 NMCLI 永久配置网络设备，以接受所有流量

您可以使用 `nmcli` 工具永久配置网络设备以接受所有流量，而不考虑 MAC 地址。

## 流程

1.

可选：显示网络接口以标识您要接收所有流量的接口：

```
ip address show
1: enp1s0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel
state DOWN group default qlen 1000
link/ether 98:fa:9b:a4:34:09 brd ff:ff:ff:ff:ff:ff
...
```

如果没有任何连接，您可以创建一个新的连接。

2.

修改网络设备以启用或禁用此属性。

- 

为 `enp1s0` 启用 `ethernet.accept-all-mac-addresses` 模式：

- # nmcli connection modify enp1s0 ethernet.accept-all-mac-addresses yes

- 为 enp1s0 禁用 accept-all-mac-addresses 模式：

- # nmcli connection modify enp1s0 ethernet.accept-all-mac-addresses no

3.

应用更改，重新激活连接：

- # nmcli connection up enp1s0

验证

- 验证是否启用了 ethernet.accept-all-mac-addresses 模式：

- # nmcli connection show enp1s0
 

```
...
 802-3-ethernet.accept-all-mac-addresses:1 (true)
```

802-3-ethernet.accept-all-mac-addresses: true 表示该模式已启用。

### 47.3. 使用 NMSTATECTL 永久配置网络设备，以接受所有流量

使用 nmstatectl 工具配置设备，以接受所有流量，而不考虑通过 Nmstate API 的 MAC 地址。Nmstate API 确保设置配置后结果与配置文件匹配。如果有任何失败，nmstatectl 会自动回滚更改以避免系统处于不正确的状态。

先决条件

- nmstate 软件包已安装。
- 用于配置设备的 enp1s0.yml 文件可用。

步骤

1.

编辑 enp1s0 连接的现有 enp1s0.yml 文件，并将以下内容添加到其中：

```

interfaces:
 - name: enp1s0
 type: ethernet
 state: up
 accept-all-mac-address: true
```

这些设置将 *enp1s0* 设备配置为接受所有流量。

2.

应用网络设置：

```
nmstatectl apply ~/enp1s0.yml
```

验证

- 验证是否启用了 802-3-ethernet.accept-all-mac-addresses 模式：

```
nmstatectl show enp1s0
interfaces:
 - name: enp1s0
 type: ethernet
 state: up
 accept-all-mac-addresses: true
...
```

802-3-ethernet.accept-all-mac-addresses: *true* 表示该模式已启用。

其他资源

- 您系统上的 **nmstatectl (8)** 手册页
- /usr/share/doc/nmstate/examples/ 目录

## 第 48 章 使用 NMCLI 镜像网络接口

网络管理员可以使用端口镜像复制从一个网络设备传输到另一个网络设备的入站和出站网络流量。在以下情况下，镜像接口的流量很有帮助：

- 要调试网络问题并调优网络流
- 要检查和分析网络流量
- 要检测入侵

### 先决条件

- 一个镜像网络流量的网络接口。

### 流程

1. 添加您要镜像网络流量的网络连接配置集：

```
nmcli connection add type ethernet ifname enp1s0 con-name enp1s0 autoconnect no
```

2. 将类型为 prio 的 qdisc 附加到带有 10: handle 的出口（传出）流量的 enp1s0：

```
nmcli connection modify enp1s0 +tc.qdisc "root prio handle 10:"
```

qdisc 设置为 prio attached without children 允许附加过滤器。

3. 为入口流量添加 qdisc，使用 ffff: handle：

```
nmcli connection modify enp1s0 +tc.qdisc "ingress handle ffff:"
```

4.

添加以下过滤器，以匹配入口和出口 qdiscs 上的数据包，并将其镜像到 `enp7s0`：

```
nmcli connection modify enp1s0 +tc.tfilter "parent ffff: matchall action mirred egress mirror dev enp7s0"
```

```
nmcli connection modify enp1s0 +tc.tfilter "parent 10: matchall action mirred egress mirror dev enp7s0"
```

`matchall` 过滤器与所有数据包匹配，`mirred` 操作会将数据包重定向到目的地。

5.

激活连接：

```
nmcli connection up enp1s0
```

## 验证

1.

安装 `tcpdump` 工具：

```
yum install tcpdump
```

2.

显示目标设备上镜像的流量 (`enp7s0`)：

```
tcpdump -i enp7s0
```

## 其他资源



[如何使用 `tcpdump` 捕获网络数据包](#) (红帽知识库)

## 第 49 章 使用 NMSTATE-AUTOCONF 自动配置使用 LLDP 的网络状态

网络设备可以使用链路层发现协议(LLDP)，来在 LAN 中公告其身份、功能和邻居。nmstate-autoconf 工具可使用此信息来自动配置本地网络接口。



### 重要

nmstate-autoconf 工具仅作为技术预览提供。红帽产品服务级别协议（SLA）不支持技术预览功能，且其功能可能并不完善，因此红帽不建议在生产环境中使用它们。这些预览可让用户早期访问将来的产品功能，让用户在开发过程中测试并提供反馈意见。

如需有关 [技术预览功能支持范围](#) 的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

### 49.1. 使用 NMSTATE-AUTOCONF 来自动配置网络接口

nmstate-autoconf 工具使用 LLDP 来识别连接到交换机的接口的 VLAN 设置来配置本地设备。

此流程假设以下场景，以及交换机使用 LLDP 广播 VLAN 设置：

- RHEL 服务器的 enp1s0 和 enp2s0 接口连接到使用 VLAN ID 100 和 VLAN 名称 prod-net 配置的交换机端口。
- RHEL 服务器的 enp3s0 接口连接到使用 VLAN ID 200 和 VLAN 名称 mgmt-net 配置的交换机端口。

然后，nmstate-autoconf 工具使用此信息来在服务器上创建以下接口：

- bond100 - enp1s0 和 enp2s0 作为端口的绑定接口。
- prod-net - 在 VLAN ID 为 100 的 bond100 上面的 VLAN 接口。

- **mgmt-net - 在 VLAN ID 为 200 的 enp3s0 上面的 VLAN 接口**

如果您将多个网络接口连接到 LLDP 用来广播同一 VLAN ID 的不同交换机的端口，则 nmstate-autoconf 会用这些接口来创建一个绑定，并在其上配置通用 VLAN ID。

## 先决条件

- **nmstate 软件包已安装。**
- **网络交换机上启用了 LLDP。**
- **以太网接口已启用。**

## 流程

1. 在以太网接口上启用 LLDP :

- a. 创建一个包含以下内容的 YAML 文件，如 ~/enable-lldp.yml :

```
interfaces:
 - name: enp1s0
 type: ethernet
 lldp:
 enabled: true
 - name: enp2s0
 type: ethernet
 lldp:
 enabled: true
 - name: enp3s0
 type: ethernet
 lldp:
 enabled: true
```

- b. 将设置应用到系统 :

```
nmstatectl apply ~/enable-lldp.yml
```

2.

### 使用 LLDP 配置网络接口：

a.

可选，启动一个空运行来显示并验证 nmstate-autoconf 生成的 YAML 配置：

```
nmstate-autoconf -d enp1s0,enp2s0,enp3s0

interfaces:
- name: prod-net
 type: vlan
 state: up
 vlan:
 base-iface: bond100
 id: 100
- name: mgmt-net
 type: vlan
 state: up
 vlan:
 base-iface: enp3s0
 id: 200
- name: bond100
 type: bond
 state: up
 link-aggregation:
 mode: balance-rr
 port:
 - enp1s0
 - enp2s0
```

b.

使用 nmstate-autoconf 根据从 LLDP 接收的信息来生成配置，并将设置应用到系统：

```
nmstate-autoconf enp1s0,enp2s0,enp3s0
```

### 后续步骤



如果您的网络中没有为接口提供 IP 设置的 DHCP 服务器，请手动配置它们。详情请查看：



[配置以太网连接](#)



[配置网络绑定](#)

### 验证

- 显示单个接口的设置：

```
nmstatectl show <interface_name>
```

## 其他资源

- 您系统上的 **nmstate-autoconf (8)** 手册页

## 第 50 章 配置 802.3 链路设置

自动协商是 IEEE 802.3u 快速以太网协议的一个特性。它以设备端口为目标，为链路上的信息交换提供速度、双工模式和流控制的最佳性能。使用自动协商协议时，您具有在以太网上进行数据传输的最佳性能。



### 注意

要最大限度地利用自动协商的性能，请在链接两端使用同样的配置。

#### 50.1. 使用 NMCLI 工具配置 802.3 链接设置

要配置以太网连接的 802.3 链接设置，请修改以下配置参数：

- **802-3-ethernet.auto-negotiate**
- **802-3-ethernet.speed**
- **802-3-ethernet.duplex**

### 流程

1.

显示连接的当前设置：

```
nmcli connection show Example-connection
...
802-3-ethernet.speed: 0
802-3-ethernet.duplex: --
802-3-ethernet.auto-negotiate: no
...
```

如果需要在有问题的时候重置参数，您可以使用这些值。

2.

设置速度和双工链路设置：

```
nmcli connection modify Example-connection 802-3-ethernet.auto-negotiate yes 802-3-ethernet.speed 10000 802-3-ethernet.duplex full
```

此命令启用自动协商，并将连接的速度设置为 10000 Mbit 全双工。

3.

重新激活连接：

```
nmcli connection up Example-connection
```

## 验证

●

使用 ethtool 工具验证以太网接口 enp1s0 的值：

```
ethtool enp1s0
Settings for enp1s0:
...
Speed: 10000 Mb/s
Duplex: Full
Auto-negotiation: on
...
Link detected: yes
```

## 其他资源

●

您系统上的 **nm-settings (5)** 手册页

## 第 51 章 DPDK 入门

数据平面开发套件(DPDK)提供库和网络驱动程序，以加快用户空间中的数据包处理。

管理员使用 DPDK，例如，在虚拟机中使用单一根 I/O 虚拟化（SR-IOV）来减少延迟并增加 I/O 吞吐量。



### 注意

红帽不支持实验性的 DPDK API。

#### 51.1. 安装 DPDK 软件包

要使用 DPDK，请安装 `dpdk` 软件包。

### 流程

- 使用 `yum` 工具安装 `dpdk` 软件包：

```
yum install dpdk
```

#### 51.2. 其他资源

- [网络适配器快速数据路径特性支持矩阵](#)

## 第 52 章 TIPC 入门

透明进程间通信(TIPC)（也称为 集群域套接字）是用于 集群范围操作的进程间通信(IPC)服务。

在高可用性和动态集群环境中运行的应用程序有特殊需要。集群中的节点数量可能会有所不同，路由器可能会失败，且出于负载均衡的考虑，功能也可以移到集群中的不同节点。TIPC 可最大程度降低应用程序开发人员处理此类问题的工作，并尽可能以正确和最佳的方式处理它们。另外，TIPC 比一般协议（如 TCP）提供效率更高且容错的通讯。

### 52.1. TIPC 的构架

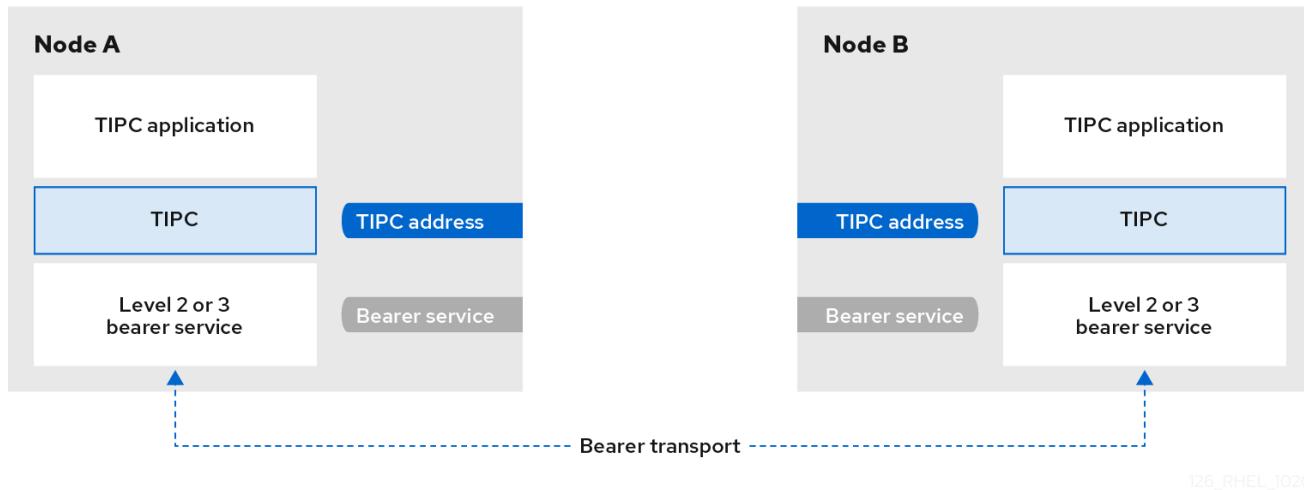
TIPC 是使用 TIPC 和数据包传输服务(bearer)的应用程序之间的一个层，横跨传输层、网络层和信令链路层。然而，TIPC 可以使用不同的传输协议作为 bearer，这样 TCP 连接就可以充当 TIPC 信号连接的 bearer。

TIPC 支持以下 bearer:

- **Ethernet**
- **InfiniBand**
- **UDP 协议**

TIPC 提供了在 TIPC 端口间可靠传送信息，这是所有 TIPC 通讯的端点。

以下是 TIPC 构架图：



## 52.2. 系统引导时载入 TIPC 模块

在使用 TIPC 协议前，您必须载入 tipc 内核模块。您可以将 Red Hat Enterprise Linux 配置为在系统引导时自动载入这个内核模块。

### 流程

1.

使用以下内容创建 /etc/modules-load.d/tipc.conf 文件：

```
tipc
```

2.

重启 systemd-modules-load 服务，以在不重启系统的情况下加载模块：

```
systemctl start systemd-modules-load
```

### 验证

- 

使用以下命令验证 RHEL 是否已载入 tipc 模块：

```
lsmod | grep tipc
tipc 311296 0
```

如果命令没有显示 tipc 模块的条目，则 RHEL 没有加载它。

### 其他资源

- 

您系统上的 **modules-load.d (5)** 手册页

### 52.3. 创建 TIPC 网络

要创建 TIPC 网络, 请在应该加入 TIPC 网络的每个主机上执行这个流程。



#### 重要

这些命令只临时配置 TIPC 网络。要在节点上永久配置 TIPC, 在脚本中使用此流程的命令, 并将 RHEL 配置为在系统引导时执行该脚本。

#### 先决条件

- 

[tipc 模块已加载。详情请查看 当系统启动时加载 tipc 模块](#)

#### 流程

1.

可选：设置一个唯一的节点身份，如 UUID 或节点的主机名：

```
tipc node set identity host_name
```

身份可以是任何由最多 16 个字母和数字组成的唯一字符串。

您不能在此步骤后设置或改变身份。

2.

添加一个 bearer。例如, 要将 Ethernet 用作介质, 并将 enp0s1 设备用作物理 bearer 设备, 请输入：

```
tipc bearer enable media eth device enp1s0
```

3.

可选：要获得冗余和更好的性能, 请使用上一步中的命令附加更多 bearer。您可以配置最多三个 bearer, 但在同一介质上不能超过两个。

4. 在应该加入 TIPC 网络的每个节点中重复前面的所有步骤。

## 验证

1. 显示集群成员的链接状态：

```
tipc link list
broadcast-link: up
5254006b74be:enp1s0-525400df55d1:enp1s0: up
```

此输出表示，节点 5254006b74be 上的 bearer enp1s0 和节点 525400df55d1 上的 bearer enp1s0 之间的链接为 up。

2. 显示 TIPC 发布表：

```
tipc nametable show
Type Lower Upper Scope Port Node
0 1795222054 1795222054 cluster 0 5254006b74be
0 3741353223 3741353223 cluster 0 525400df55d1
1 1 1 node 2399405586 5254006b74be
2 3741353223 3741353223 node 0 5254006b74be
```

- 服务类型为 0 的两个条目表示两个节点是这个集群的成员。
- 服务类型为 1 的条目代表内置的拓扑服务跟踪服务。
- 服务类型为 2 的条目显示从发布节点看到的链接。范围限制 3741353223 表示十进制格式的对等端点的地址（基于节点身份的唯一 32 位哈希值）。

## 其他资源

- 您系统上的 [tipc-bearer \(8\)](#) 和 [tipc-namespace \(8\)](#) 手册页

### 52.4. 其他资源

- 红帽建议使用其他 **bearer** 级别协议来根据传输介质加密节点之间的通信。例如：
  - **MACsec**：请参阅 [使用 MACsec 来加密第 2 层流量](#)
  - **IPsec**：请参阅 [配置带有 IPsec 的 VPN](#)
- 有关如何使用 TIPC 的示例，请使用 `git clone git://git.code.sf.net/p/tipc/tipc/tipcutils` 命令克隆上游 GIT 存储库。该存储库包含使用 TIPC 功能的演示和测试程序的源代码。请注意，这个存储库不是红帽提供的。
- `/usr/share/doc/kernel-doc-<kernel_version>/Documentation/output/networking/tipc.html` 由 `kernel-doc` 软件包提供。

## 第 53 章 使用 NM-CLOUD-SETUP 在公有云中自动配置网络接口

通常，虚拟机(VM)只有一个可被 DHCP 配置的接口。但是，DHCP 无法使用多个网络实体，如接口、IP 子网和 IP 地址来配置虚拟机。另外，您无法在虚拟机实例运行时应用设置。要解决这个运行时配置问题，`nm-cloud-setup` 工具会自动从云服务提供商的元数据服务器检索配置信息，并更新主机的网络配置。工具自动获取一个接口上的多个网络接口、多个 IP 地址或 IP 子网，并帮助重新配置正在运行的虚拟机实例的网络。

### 53.1. 配置和预部署 NM-CLOUD-SETUP

要在公有云中启用和配置网络接口，请运行 `nm-cloud-setup` 作为计时器和服务。



#### 注意

在 Red Hat Enterprise Linux On Demand 和 AWS 黄金镜像上，`nn-cloud-setup` 已启用，无需任何操作。

#### 先决条件

- 存在网络连接。
- 连接使用 DHCP。

默认情况下，NetworkManager 会创建一个使用 DHCP 的连接配置文件。如果因为您在 `/etc/NetworkManager/NetworkManager.conf` 中设置了 `no-auto-default` 参数而没有创建配置文件，请手动创建此初始连接。

#### 流程

1. 安装 `nm-cloud-setup` 软件包：
 

```
yum install NetworkManager-cloud-setup
```
2. 为 `nm-cloud-setup` 服务创建并运行 管理单元文件：
  - a. 使用以下命令开始编辑管理单元文件：
 

```
sudo nmcli connection modify <connection> ipv4.method manual
```

```
[root@rhel8 ~]# systemctl edit nm-cloud-setup.service
```

```
systemctl edit nm-cloud-setup.service
```

显式启动服务或重启系统，以使配置设置生效是非常重要的。

b.

使用 `systemd` 管理单元文件来在 `nm-cloud-setup` 中配置云提供商。例如，要使用 Amazon EC2，请输入：

```
[Service]
Environment=NM_CLOUD_SETUP_EC2=yes
```

您可以设置以下环境变量来启用您所使用的云提供商：

- 用于 Alibaba Cloud (Aliyun) 的 `NM_CLOUD_SETUP_ALIYUN`
- 用于 Microsoft Azure 的 `NM_CLOUD_SETUP_AZURE`
- 用于 Amazon EC2(AWS)的 `NM_CLOUD_SETUP_EC2`
- 用于 Google Cloud Platform(GCP)的 `NM_CLOUD_SETUP_GCP`

c.

保存文件并退出编辑器。

3.

重新载入 `systemd` 配置：

```
systemctl daemon-reload
```

4.

启用并启动 `nm-cloud-setup` 服务：

```
systemctl enable --now nm-cloud-setup.service
```

5.

启用并启动 `nm-cloud-setup` 计时器：

```
systemctl enable --now nm-cloud-setup.timer
```

## 其他资源

- 您系统上的 `nm-cloud-setup (8)` 手册页
- [配置以太网连接](#)

### 53.2. 了解 RHEL EC2 实例中 IMDSv2 角色和 NM-CLOUD-SETUP

Amazon EC2 中的实例元数据服务(IMDS)允许您管理访问正在运行的 Red Hat Enterprise Linux (RHEL) EC2 实例的实例元数据的权限。RHEL EC2 实例使用 IMDS 版本 2 (IMDSv2)，一个面向会话的方法。通过使用 `nm-cloud-setup` 工具，管理员可以重新配置网络，并自动更新正在运行的 RHEL EC2 实例的配置。`nm-cloud-setup` 工具使用 IMDSv2 令牌处理 IMDSv2 API 调用，而无需用户干预。

- IMDS 运行在本地链路地址 169.254.169.254 上，提供对 RHEL EC2 实例上原生应用程序的访问。
- 为应用程序和用户的每个 RHEL EC2 实例指定并配置了 IMDSv2 后，您无法再访问 IMDSv1。
- 通过使用 IMDSv2，RHEL EC2 实例可在不使用 IAM 角色的情况下维护元数据，同时通过 IAM 角色保持可访问。
- 当 RHEL EC2 实例引导时，`nmn-cloud-setup` 工具自动运行，以获取使用 RHEL EC2 实例 API 的 EC2 实例 API 访问令牌。



#### 注意

使用 IMDSv2 令牌作为 HTTP 标头，来检查 EC2 环境的详情。

## 其他资源



您系统上的 **nm-cloud-setup (8)** 手册页