第 3 章

物理硬件与软件技术选型

3.1 物理硬件的分析与比较

所有的软件都需要硬件作为基础，都需要一个基础硬件运行平台，在做硬件选型时，应该结合软件的特点，在满足软件需求的情况下，选择具有优势的硬件，方可达到相得益彰的效果。

3.1.1 网络硬件设备

软件的核心功能是建立链路，实现匿名路由功能，因此，设备需要支持路由功能，并且可以支持连接多个外部设备，即具有多个网口。通过初步调研和分析，如表 3-1 所示的几种设备符合要求并具有实现功能的潜力。

表 3-1 网络设备特性分析

设备名称	原有功能	二次开发	产品支持	设备价格
路由器	支持无线连接和网线连接	支持 Python 开发，某些设备支持其中的一些 IPSec/OpenVPN/L2TP/PPTP/DMVPN 等单跳 VPN 协议	ORB305-4G 4GDTU	千元内
交换机	支持 VLAN 配置，多网络连接	支持 SDK 二次开发	ISM7112G-4GF-8GT MES7106G-2XGF-4GT	千元内
定制设备	支持 LINUX 运行环境和硬件配置	支持自由定制开发	J4125 J1900	千元内

3.1.2 方案评估

根据网络硬件的特点，结合需要实现软件功能，进行方案假设与评估，以及开发的难度，选择最适合完成开发的网络硬件，网络硬件方案假设如表 3-2 所示。

表 3-2　网络硬件设备方案假设

产品名称	设备特点	方案假设
工业路由器	支持二次开发且带有多种 VPN 协议的路由器，支持多网口、端口异常告警、多电源冗余备份，成本相对不算太高，目前根据了解到的资料，配置的是单跳 VPN 节点，没有多个节点模式	基于现有的 VPN 协议进行实现，主要开发一个 Web 管理，管理所有的数据流量，主要实现节点管理和链路管理，让数据经过多次 VPN 节点转发之后出去
工业交换机	也有支持二次开发的交换机设备，但是大都没有自带 VPN 协议支持，优势是原生支持 VLAN	基于交换机的系统进行二次开发，相对来讲，难度比路由器大一点，二次开发的数据流量控制对原有系统的功能影响或冲突需要进一步确定了解
工业网关	主要查阅的资料是基于 PLC 模块进行的开发，也支持二次开发，具有网络流量接入功能	对原有数据传输方式进行控制，控制数据的传输方向，把数据根据开发的链路和节点进行传输。需要考虑原有数据传输方式如何进行改变和控制的问题
定制设备	硬件参数根据开发需要进行定制，对内存、CPU、硬盘可以实现按需配置，例如日志存储周期、其他信息统计等	可以安装常规等 Linux 系统，基于 Iptables 和 Route 进行流量控制，通过 Python、Go 开发 Web 应用管理，调用系统命令实现一系列的网络和路由管理

通过对方案的评估与分析，任何一种网络硬件都需要进行二次开发，有些硬件设备，例如路由器、交换机，本身只支持一跳的 VPN 路由，需要进行改进，这相当于需要进行路由的重定向和同时支持多种 VPN 的拨号，结合设备的硬件参数普遍很低，需要进行裁剪或者嵌入式开发的，难度较大。有些是基于 C 语言进行的开发的，例如网关设备，其本身是嵌入式开发的，如果要支持多种 VPN 协议连接，则需要进行 VPN 裁剪适配，如果要改变数据传输路由，则需要重写相关的路由模块，但是硬件的配置较低，要做到真正适配和流畅运行，难度较大。定制设备的硬件配置相对较高，能够支持完整的 Linux 系统运行，也就意味着可以选择多种开发语言，大幅度降低了开发门槛，并且硬件设备价格接近，有成熟的开源软件可以使用，可以达到对网络设备开发的更多可控，综合考虑开发成本和硬件设备，本书选择以定制设备为例，从零开始开发，实现一款支持匿名链路的网络路由设备。

3.2　主流开源操作系统简介

Linux 系统作为一个开源性的操作系统，受到不少程序员的青睐，衍生出可满足各种不同需要的版本，可以根据自身需要进行修改设置，比起微软更受企业欢迎，大部分网站采用的系统是 Linux 系统，它主要有以下特点。

1. 大量的可用软件及免费软件

Linux 系统上有着大量的可用软件，并且绝大多数是免费的，例如声名赫赫的 Apache、

Samba、PHP、MySQL 等，构建成本低廉，这是 Linux 被众多企业青睐的原因之一。当然，这和 Linux 出色的性能是分不开的，否则节约成本就没有任何意义。

但不可否认的是，Linux 在办公应用和游戏娱乐方面的软件相比 Windows 系统还很匮乏，所以玩游戏、看影片等大多采用 Windows，至于 Linux，主要把它用在擅长的服务器领域。

2. 良好的可移植性及灵活性

Linux 系统有良好的可移植性，它支持大部分 CPU 平台，这使它便于裁剪和定制。可以把 Linux 放在 U 盘、光盘等存储介质中，也可以在嵌入式领域广泛应用。

如果希望不进行安装就体验 Linux 系统，则可以在网上下载一个 Live DVD 版的 Linux 镜像，刻成光盘放入光驱或者用虚拟机软件直接载入镜像文件，将 CMOS/BIOS 设置为光盘启动，系统就会自动载入光盘文件，启动后便可进入 Linux 系统。

3. 优良的稳定性和安全性

著名的黑客埃里克·雷蒙德(Eric S. Raymond)有一句名言："足够多的眼睛，就可让所有问题浮现。"举个例子，假如笔者在演讲，台下人山人海，明哥中午吃饭不小心，有几个饭粒粘在衣领上了，分分钟就会被大家发现，因为看的人太多了；如果台下就两三个人且离得很远，就算明哥衣领上有一大块油渍也不会被发现。

Linux 开放源代码，将所有代码放在网上，全世界的程序员都看得到，有什么缺陷和漏洞，很快就会被发现，从而成就了它的稳定性和安全注。

4. 支持绝大多数网络协议及开发语言

UNIX 系统是与 C 语言、TCP/IP 一同发展起来的，而 Linux 是一种类 UNIX 系统，C 语言又衍生出了现今主流的语言 PHP、Java、C++等，而哪一个网络协议与 TCP/IP 无关呢？所以，Linux 对网络协议和开发语言的支持很好。

Linux 是一套免费使用和自由传播的类 UNIX 操作系统，是一个基于 POSIX 和 UNIX 的多用户、多任务、支持多线程和多 CPU 的操作系统。它能运行主要的 UNIX 工具软件、应用程序和网络协议。它支持 32 位和 64 位硬件。Linux 继承了 UNIX 以网络为核心的设计思想，是一个性能稳定的多用户网络操作系统。不同发行版本的 Linux 系统有着细微的区别，常用的几种 Linux 系统如下。

1) Debian 与 Ubuntu 对比分析

Ubuntu 是基于 Debian 开发的，可以简单地认为 Ubuntu 是 Debian 的功能加强版。与 Debian 相比，Ubuntu 提供了更人性化的系统配置，更强大的系统操作及比 Debian 更激进的软件更新。Ubuntu 与 Debian 比较，可以认为 Debian 更保守一些，Ubuntu 对新手友好度更高，上手更容易。用过 Ubuntu 的人都会体会到它的易用，反之如果用过 Ubuntu 再换到别的系统则会觉得不适应，Ubuntu 真的很方便。

Ubuntu 普通版本只提供 18 个月的技术支持，过期则不再提供技术支持。LTS 服务器版本提供长达 5 年的技术支持。例如，Ubuntu 10.10 是个普通版，现在已经过了支持周期了。使用时会发现安装不了任何软件，因为 10.10 的软件已经从 Ubuntu 软件源中被移除

了，所以建议选择 LTS 版，提供长达 5 年的技术支持，可以确保在今后相当长的一段时间内服务器可以继续收到系统升级补丁及可用的软件。

2）RHEL 和 CentOS 对比分析

RHEL 跟 CentOS 差别不大。RHEL 是付费操作系统，可以免费使用，但是如果要使用 RHEL 的软件源并且想得到技术支持，则要像 Windows 那样付费。符合开源精神，免费使用，但服务收费。

CentOS 是 RHEL 的开源版本。一般在 RHEL 更新之后，CentOS 会把代码中含有 RHEL 专利的部分去掉，同时 RHEL 中包含的种种服务器设置工具也一起去掉，然后重新编译成 CentOS。从某种意义上讲，CentOS 几乎可以被看成 RHEL，这两个版本的 rpm 包是可以通用的。

不管什么发行版本的 Linux 都是基于 Linux 内核进行了很多扩展，在每个产品和业务场景下，可以根据需要进行选择，在本次开发系统中，可以选中 CentOS 系统，并对系统进行内核裁剪，去掉不需要的服务，节省内存和硬盘空间，提高运行效率，例如嵌入式开发，采用的大都是裁剪之后的 Linux，并加入了一些定制应用，完成特定的业务场景，提高运行效率。

3.3　流行的 VPN 技术调研分析

市面上常见的 VPN 技术有几十种，但是每种 VPN 之间也存在一些差异，结合业务场景和需要实现的功能，对主流的几种 VPN 进行分析对比，主要从建立连接的验证方式、支持的流量协议、支持的操作系统平台、是否容易部署等方面进行比较分析，选择合适的 VPN 技术作为隧道连接技术，本书采用 3 种不同的 VPN 进行实现，以便在一条链路上采用不同的 VPN 协议提高数据的安全性。常用的几种 VPN 技术特点对比分析如表 3-3 所示。

表 3-3　常用 VPN 特性分析

VPN 名称	验证方式	支持流量	支持系统
OpenVPN	公钥私钥（加账号和密码）方式验证	TCP/UDP/ICMP/SNMP/SMTP/IGMP	Solaris、Linux、OpenBSD、FreeBSD、NetBSD、macOS 与 Windows 2000/XP/Vista
Wireguard	公钥私钥（加账号和密码）方式验证	全类型	Linux、Windows、macOS、BSD、iOS 和 Android
Ocserv	账号和密码方式认证	全类型	多平台
StrongSwan	公钥私钥（加账号和密码）方式验证	全类型	多平台

3.3.1　路由器固件

主流路由器固件有 dd-wrt、tomato、openwrt、padavan 四类，对比一个单一的、静态的系

统，OpenWrt 的包管理提供了一个完全可写的文件系统，从应用程序供应商提供的选择和配置，并允许自定义的设备，以适应任何应用程序。对于开发人员，OpenWrt 是使用框架来构建应用程序的，而无须建立一个完整的固件来支持。

3.3.2 VPN 的分类标准

1. 按 VPN 的协议分类

VPN 的隧道协议主要有 3 种，即 PPTP、L2TP 和 IPSec，其中 PPTP 和 L2TP 工作在 OSI 模型的第 2 层，又称为二层隧道协议；IPSec 是第 3 层隧道协议。

2. 按 VPN 的应用分类

(1) Access VPN(远程接入 VPN)：客户端到网关，使用公网作为骨干网在设备之间传输 VPN 数据流量。

(2) Intranet VPN(内联网 VPN)：网关到网关，通过公司的网络架构连接来自同公司的资源。

(3) Extranet VPN(外联网 VPN)：与合作伙伴企业网构成 Extranet，将一个公司与另一个公司的资源进行连接。

3. 按所用的设备类型进行分类

网络设备提供商针对不同客户的需求，开发出不同的 VPN 网络设备，主要为交换机、路由器和防火墙。

(1) 路由器式 VPN：路由器式 VPN 部署较容易，只要在路由器上添加 VPN 服务即可。

(2) 交换机式 VPN：主要应用于连接用户较少的 VPN 网络。

(3) 防火墙式 VPN：防火墙式 VPN 是最常见的一种实现方式，许多厂商提供这种配置类型。

4. 按照实现原理划分

(1) 重叠 VPN：此 VPN 需要用户自己建立端节点之间的 VPN 链路，主要包括 GRE、L2TP、IPSec 等众多技术。

(2) 对等 VPN：由网络运营商在主干网上完成 VPN 通道的建立，主要包括 MPLS、VPN 技术。

3.3.3 开源 VPN 解决方案

1. Openswan

Openswan 是 Linux 的 IPSec 实现，支持大多数与 IPSec 相关的扩展(包括 IKEv2)。从 2005 年初开始，它就被广泛地认为是 Linux 用户的首选 VPN 软件。根据正在运行的 Linux 版本的不同，Openswan 可能已经被嵌入发行版中，也可以直接从它的站点下载源代码。

2. Tcpcrypt

Tcpcrypt 协议是一种独特的 VPN 解决方案，它不需要配置或对应用程序进行更改，也不需要在网络连接中进行过多设置。Tcpcrypt 使用一种称为“机会式加密”的操作方式。这意味着如果连接的另一端支持与 Tcpcrypt 通信，则通信将被加密，否则它可以被视为明文。

虽然这不够完美，但该协议已经经历了许多强大的更新，使其可以更好地抵御被动和主动攻击。尽管不建议将 Tcpcrypt 作为全公司范围的解决方案，但对于处理不那么敏感信息的员工和分支机构来讲，它可以作为一个极好的、易于实现的解决方案。

3. Tinc

Tinc 是使用 GNU 通用公共许可证授权的自由软件。Tinc 与列表中的其他 VPN（包括 OpenVPN 协议）的区别在于它包含了各种独特特性，包括加密、可选压缩、自动网格路由和易于扩展。这些特性使 Tinc 成为那些想要在众多相隔甚远的小型网络中创建 VPN 的企业的理想解决方案。

4. SoftEther VPN

SoftEther（软件以太网的缩写）VPN 是迄今为止市场上最强大和用户友好的多协议 VPN 软件之一。作为 OpenVPN 的理想替代品，SoftEther VPN 为 OpenVPN 服务器提供了克隆功能，允许无缝地从 OpenVPN 迁移到 SoftEther VPN。SoftEther 令人印象深刻的安全标准和功能被认为可以与 NordVPN 等市场领先企业相媲美，使其成为开源巨头之一。

SoftEther 还兼容 L2TP 和 IPSec 协议，支持用户定制。此外，SoftEther VPN 还被证明比 OpenVPN 更快，拥有更好的浏览体验。SoftEther 的主要缺点是它在兼容性方面落后于其他方案，然而，这个问题的主要原因是 SoftEther 协议的相对新颖性，并且随着时间的推移，可能会看到越来越多的平台支持 SoftEther。

5. OpenConnect

考虑到 OpenConnect 是为支持思科 AnyConnect SSL VPN 而创建的 VPN 客户端，可能会惊讶于在列表中看到此软件，但是，需要注意的是，OpenConnect 与 Cisco 或 Pulse Secure 并没有正式的关联。它只是与那些设备兼容。

事实上，在对思科客户进行测试之后，OpenConnect 被发现存在大量安全漏洞，OpenConnect 正着手整顿。如今，OpenConnect 已经解决了所有思科客户端的缺陷，使它成为任何 Linux 用户的思科替代品之一。

6. Libreswan

经过 15 年的积极开发，Libreswan 成为现代市场上最好的开源 VPN 替代品之一。Libreswan 目前支持最常见的 VPN 协议，即 IPSec、IKEv1 和 IKEv2。与 Tcpcrypt 一样，Libreswan 基于机会加密进行操作，这使其容易受到主动攻击，然而，大量的安全功能和活跃的社区开发人员使 Libreswan 成为低中级加密要求的理想选择。

7. StrongSwan

由通信安全教授、瑞士应用科学大学互联网技术与应用研究所所长 Andreas Steffen 负

责维护的 StrongSwan 在 VPN 社区中独树一帜，它提供了卓越的加密标准和简单的配置，以及支持大型复杂 VPN 网络的 IPSec 策略。

8. Algo

Algo 是一款极简主义的 VPN 创建工具，面向需要经常移动的用户。因为它的设计主要是为了简单和保密，所以 Algo 是不可扩展的，不能用于逃避审查、地理隔离等任务。Algo 只支持 Wireguard 和 IKEv2 协议，不需要 OpenVPN 或任何其他客户端应用。设置起来既简单又快捷，所以如果只需一个安全的代理，则 Algo 是一个不错的选择。

9. Streisand

Streisand 可以被称为是一个更强大和灵活的 Algo，然而，它并不支持 IKEv2，但可以使用它轻松地绕过审查，而且它的设置几乎不需要任何专业知识。它支持 OpenSSH、OpenConnect、L2TP、OpenVPN、Shadowsocks、Torbridge、WireGuard 和 Stunnel，需要安装的客户端应用程序取决于决定实现哪个协议。

10. PriTunl

PriTunl 是一款开源软件，能够使用它创建一个云 VPN，该云 VPN 具有安全加密、复杂的站点到站点链接、网关链接及通过 Web 界面远程访问本地网络中的用户等功能。PriTunl 拥有多达 5 个认证层、可定制的插件系统、跨平台的官方客户端、对 OpenVPN 客户端和 AWS VPC 网络的支持，并且易于设置。

11. OpenVPN

OpenVPN 是目前最流行的 VPN 解决方案之一。它与同名的协议一起工作，甚至可以使用它穿越 NAT 防火墙。它支持 TCP 和 UDP 传输，多种加密方法，是完全可定制的。不过，应该注意，需要使用客户端应用程序。OpenVPN 的使用相对其他几款工具而言有些复杂，但不要担心，因为有很多指南和一个社区帮助从初学者过渡到专业用户。

12. WireGuard

在列出 OpenVPN 和 StrongSwan 等工具之后，是时候推出一个更容易使用的 VPN 解决方案了。WireGuard 是一个多平台工具，可以轻松地使用它的同名协议部署 VPN。再加上它对 IPv4 和 IPv6 的支持，它最突出的特性是加密密钥路由——一个将公共密钥与隧道中的 IP 地址列表相关联的特性。WireGuard 的目标是成为最简单、最安全、最容易使用的 VPN 解决方案，很多用户已经在使用。

13. VyOS

VyOS 不同于其他产品，因为它是一个成熟的网络 Linux 操作系统，专门为路由器和防火墙开发。它具有 Web 代理和站点过滤、针对 IPv4 和 IPv6 的站点到站点 IPSec、针对站点到站点和远程访问的 OpenVPN 及对动态路由协议和 CLI 的全面支持及其他高级路由特性。VyOS 是从头开始构建的，提供优秀的 VPN 特性，可以根据自己的喜好定制它们。

14. Freelan

Freelan 是一款免费、开源、多平台、对等的 VPN 软件，通过互联网抽象 LAN，除了使用它为用户提供访问私有网络的特权之外，还可以使用以网络拓扑来创建 VPN 服务。

Freelan 是用 C 和 C++编写的，侧重于安全性、性能和稳定性。作为 VPN 软件，用户所需要做的就是安装和配置它，并允许它在后台运行。如果想建立一个能够匿名上网的网络代理，则可以到对应的社区查询相关的资料。

15. Outline

Outline 是 Jigsaw 的网络安全部门发布的一个项目，其目的是允许用户在 digitalocean 上创建一个 VPN 服务器并授权访问它。Outline 本身并不是 VPN，它依赖于 Shadowsocks 协议（用于重定向互联网流量的加密 socks5 代理）。它有一个漂亮的 GUImanager 应用程序，易于使用，用户可以从中设置配置和选择服务。

3.4　网络与安全的核心工具 iptables

3.4.1　iptables 介绍

iptables 是在 Linux 内核里配置防火墙规则的用户空间工具，它实际上是 Netfilter 框架的一部分。可能因为 iptables 是 Netfilter 框架里最常见的部分，所以这个框架通常被称为 iptables，iptables 是 Linux 从 2.4 版本引入的防火墙解决方案。

1. iptables 规则链的类型

iptables 的规则链分为 3 种：输入、转发和输出。

（1）输入：这条链用来过滤目的地址是本机的连接。例如，如果一个用户试图使用 SSH 登录到 PC/服务器，iptables 则会首先将其 IP 地址和端口匹配到 iptables 的输入链规则。

（2）转发：这条链用来过滤目的地址和源地址都不是本机的连接。例如，路由器收到的绝大多数数据需要转发给其他主机。如果系统没有开启类似于路由器的功能，如 NATing，就不需要使用这条链。

（3）输出：这条链用来过滤源地址是本机的连接。例如，当尝试 ping howtogeek.com 时，iptables 会检查输出链中与 ping 和 howtogeek.com 相关的规则，然后决定允许还是拒绝连接请求。

当 ping 一台外部主机时，看上去好像只是输出链在起作用，但是，外部主机返回的数据要经过输入链的过滤。当配置 iptables 规则时，应牢记许多协议需要双向通信，所以需要同时配置输入链和输出链。人们在配置 SSH 时通常会忘记在输入链和输出链都配置它。

2. 链的默认行为

在配置特定的规则之前，应配置这些链的默认行为。换句话说，当 iptables 无法匹配现存的规则时，想让它做出何种行为。

可以运行如下的命令来显示当前 iptables 对无法匹配的连接的默认动作：

```
iptables -L | grep policy
```

通常情况下，系统在默认情况下接收所有的网络数据。这种设定也是 iptables 的默认配置。接收网络连接的配置命令如下：

```
iptables -- policy INPUT ACCEPT
iptables -- policy OUTPUT ACCEPT
iptables -- policy FORWARD ACCEPT
```

在使用默认配置的情况下，也可以添加一些命令来过滤特定的 IP 地址或端口号。

如果想在默认情况下拒绝所有的网络连接，然后在其基础上添加允许的 IP 地址或端口号，则可以将默认配置中的 ACCEPT 变成 DROP，命令如下所示。这对于一些含有敏感数据的服务器来讲是极有用的。通常这些服务器只允许特定的 IP 地址访问它们：

```
iptables -- policy INPUT DROP
iptables -- policy OUTPUT DROP
iptables -- policy FORWARD DROP
```

对特定连接的配置，可对特定的 IP 地址或端口做出设定。主要介绍 3 种最基本和常见的设定。

(1) Accept：接收所有的数据。

(2) Drop：丢弃数据。应用场景：当不想让数据的来源地址意识到系统的存在时，这是最好的处理方法。

(3) Reject：不允许建立连接，但是返回一个错误回应。应用场景：不想让某个 IP 地址访问系统，但又想让访问者知道防火墙阻止了其访问。

3.4.2 ipset 介绍

1. 基本的 ipset 用法

ipset 是 iptables 的扩展，它允许创建匹配整个地址 sets(地址集合)的规则，而不像普通的 iptables 链那样(线性的存储和过滤)，IP 集合存储在带索引的数据结构中，这种结构即使集合比较大也可以进行高效查找。

除了一些常用的情况，例如阻止一些危险的主机访问本机，从而减少系统资源占用或网络拥塞，ipset 也具备一些新防火墙设计方法，并简化了配置。

例如 Web 服务(端口一般在 80，HTTPS 端口一般在 443)，可以使用 0.0.0.0/0 来阻止所有的 IP 地址，命令如下：

```
iptables -A INPUT -s 0.0.0.0/0 -p tcp --dport 80 -j DROP
iptables -A INPUT -s 0.0.0.0/0 -p tcp --dport 443 -j DROP
```

阻止所有访问 Web 服务器的 IP 地址，然后将指定的 IP 添加到白名单，例如添加 1.2.3.4，命令如下：

```
iptables -A INPUT -s 1.2.3.4 -p tcp --dport 80 -j ACCEPT
```

如果允许某个网段下的所有 IP 都可以访问，例如 1.2.3.[0-255]，则使用的命令如下：

```
iptables -A INPUT -s 1.2.3.0/24 -p tcp --dport -j ACCEPT
```

总之不管是阻止所有的服务还是只阻止指定的服务，都可以先将默认的规则设置为所有 IP 都不可访问，然后手动将 IP 地址添加到白名单。

ipset 创建 set，命令如下：

```
ipset create banip hash:net
```

这里是创建一个名为 banip 的集合，以哈希方式存储，存储内容是 IP/段地址，然后在这个集合里面添加需要封禁的 IP 或者 IP 段，命令如下：

```
ipset add banip 4.5.6.7                    //单个 IP 地址

ipset add banip 4.5.6.0/24                 //IP 段
```

这里需要注意一点，现在添加的 IP 集合 banip 当服务器重启后会丢失，因为集合是临时的，所以需要执行以下命令进行持久化保存：

```
service ipset save
```

执行之后，会被保存在/etc/sysconfig/ipset 中，下次重启时会自动调用，以后要加 IP 时直接加在 /etc/sysconfig/ipset 文件内就可以了。

接下来还有最后一步，在 iptables 里添加一个规则，以此来调用这个集合，命令如下：

```
iptables -I INPUT -m set -match-set banip src -p tcp -destination-port 80 -j DROP
```

参数-m set -match-set banip src 用于实现调用 banip 集合，封禁的是 80 端口，只要在 banip 集合里的 IP 全部无法访问 80 端口。重启 iptables 后生效，命令如下：

```
service iptables restart
```

至此，全部配置就已经完成了，添加到 banip 集合里的 IP 无法访问 80 端口，并且后续添加 IP 时不需要重启 iptables，可立即生效。

2. 更多的 ipset 用法

(1) 存储类型的使用。前面例子中的 banip 集合是以哈希方式存储 IP 和 IP 段地址的，也就是以网络段为哈希的键。除了 IP 地址，还可以是 IP 段、端口号(支持指定 TCP/UDP)、MAC 地址、网络接口名称，或者上述各种类型的组合。

例如指定 hash：ip,port 就是 IP 地址和端口号共同作为哈希的键。查看 ipset 的帮助文档可以看到它支持的所有类型。

下面以两个例子进行说明。

创建 ipset 集合，IP 为 key，并进行测试，命令如下：

```
ipset create banip hash:ip
ipset add banip 6.7.8.9
ipset test banip 1.2.3.2
```

执行 ipset test banip 1.2.3.2 就会得到结果：

```
1.2.3.2 is NOT in set banip
```

注意，如果在 hash：ip 里添加 IP 段，则会被拆分为这个 IP 段里所有的独立 IP 进行存储，如果是 IP 和 IP 段混合存储，则建议使用 hash：net。

创建 ipset 集合，ip 和 port 为 key，并进行测试，命令如下：

```
ipset create banip hash:ip,port
ipset add banip 3.4.5.6,80
ipset add banip 5.6.7.8,udp:53
ipset add banip 1.2.3.4,80-86
```

第 2 条命令添加的 IP 地址为 3.4.5.6，端口号是 80。如果没有注明协议，默认就是 TCP，下面一条命令则指明了使用 UDP 的 53 端口。最后一条命令指明了一个 IP 地址和一个端口号范围，这也是合法的命令。

(2) 自动过期和解封的使用。ipset 支持 timeout 参数，这就意味着，如果一个集合作为黑名单被使用，通过 timeout 参数，就可以到期自动从黑名单里删除内容。添加自动过期的命令如下：

```
ipset create banip hash:ip timeout 300
ipset add banip 1.2.3.4
ipset add banip 6.6.6.6 timeout 60
```

上面第 1 条命令创建了名为 banip 的集合，后面添加了 timeout 参数，值为 300，往集合里添加条目的默认 timeout 时间就是 300。第 3 条命令在向集合添加 IP 时指定了一个不同于默认值的 timeout 值 60，由此可知这一条就会在 60s 后被自动删除。

隔几秒执行一次 ipset list banip 可以看到这个集合里的 timeout 一直在随着时间的变化而变化，标志着它们在多少秒之后会被删除。

如果要重新为某个条目指定 timeout 参数，则要使用-exist 选项。

```
ipset -exist add banip 1.2.3.4 timeout 100
```

这样 1.2.3.4 这一条数据的 timeout 值就变成了 100，如果在这里将其设置为 300，则它的 timeout(存活时间)会重新变成 300。

如果在创建集合时没有指定 timeout，则之后添加的条目不支持 timeout 参数，执行 add 会收到报错。如果想要默认条目不过期(自动删除)，并且需要在添加某些条目时加上

timeout 参数，则可以在创建集合时将 timeout 指定为 0。

ipset 集合支持更大条目，hashsize 和 maxelem 参数分别指定了创建集合时初始的哈希大小和最大存储的条目数量。创建命令如下：

```
ipset create banip hash:ip hashsize 4096 maxelem 1000000
ipset add banip 3.4.5.6
```

这样就创建了名为 banip 的集合，初始哈希大小是 4096，如果满了，则这个哈希会自动扩容为之前的两倍。最大能存储的数量是 100 000 个。如果没有指定，则 hashsize 的默认值为 1024，maxelem 的默认值为 65 536。

ipset 常用命令如下：

```
ipset del banip x.x.x.x              #从 banip 集合中删除内容
ipset list banip                     #查看 banip 集合的内容
ipset list                           #查看所有集合的内容
ipset flush banip                    #清空 banip 集合
ipset flush                          #清空所有集合
ipset destroy banip                  #销毁 banip 集合
ipset destroy                        #销毁所有集合
ipset save banip > 1.txt             #将 banip 集合内容输出到 1.txt
ipset save > 1.txt                   #将所有集合内容输出到 1.txt
ipset restore < 1.txt                #根据 1.txt 内容恢复集合内容
service ipset save                   #执行 add 和 del 这类添加和删除操作后都需要保存
service ipset restart                #重启 ipset
```

限制来自某一 IP 的并发访问，环境如下。

A 的 IP：192.168.31.158。

B 的 IP：192.168.31.207。

在 B 上执行 ab 命令，模拟大量请求，命令如下：

```
ab -n 10000 -c 20 http://192.168.31.158/test.html
```

完成后，到 A 中查看负载状况，执行 w 命令查看详情。

发现 A 的压力太大，得限制 B 了，执行 iptables 命令如下：

```
iptables -I INPUT -p tcp --dport 80 -s 192.168.31.207 -m connlimit --connlimit-above
10 -j REJECT
```

再到 B 中执行之前的 ab 命令，命令如下：

```
ab -n 10000 -c 20 http://192.168.31.158/test.html
```

发现请求已经被拒绝了。如果把参数-c 的值改为 9，就可以正常执行了。

为了便于理解，这个 iptables 命令可以分为几部分命令说明：

```
iptables
    -I INPUT
    -p tcp --dport 80 -s 192.168.31.207
    -m connlimit --connlimit-above 10
    -j REJECT
```

-I INPUT：表示要插入一条 INPUT 链的规则。

-p tcp --dport 80 -s 192.168.31.207：针对来自 192.168.31.207 这个 IP 对于本机 80 端口的 TCP 请求。

-m connlimit --connlimit-above 10：表示匹配条件，并发数大于 10 时成立。

-j REJECT：满足条件后要执行的动作，即拒绝。

iptables 删除除一个 IP 之外的所有传入 ICMP 请求。iptables 对命令运行的顺序很敏感。如果规则匹配，它则不会继续检查更多规则，它只是服从那个规则。如果先设置 drop，则接受规则永远不会被测试。通过源 IP 设置特定接受规则，然后将更一般的策略设置为丢弃将影响预期行为。错误的配置命令如下：

```
iptables -A INPUT -s x.x.x.x -p ICMP --icmp-type 8 -j ACCEPT
iptables -A INPUT -p ICMP --icmp-type 8 -j DROP
```

上面的配置命令导致 ICMP 请求永远被全部丢弃，正确的配置命令如下：

```
iptables -A INPUT -p ICMP --icmp-type 8 -j DROP
iptables -A INPUT -s x.x.x.x -p ICMP --icmp-type 8 -j ACCEPT
```

3.5 非常强大的网络管理工具 ip 命令

在以前的 Linux 系统版本中，可使用 ifconfig 命令查看 IP 地址等信息，但是 ifconfig 已经不再被维护了，并在近几年的 Linux 版本中已经被弃用。ifconfig 命令已被 ip 命令替换。ip 命令有点类似于 ifconfig 命令，但它更强大，附加了更多的功能。ip 命令可以执行一些网络相关的任务，是 ifconfig 不能操作的。

ip 命令可以显示或操作路由、网络设备、设置路由策略和通道。此命令的适用范围：RHEL、Ubuntu、CentOS、SUSE、openSUSE、Fedora。

3.5.1 使用语法

ip [选项] OBJECT COMMAND [help]

OBJECT 对象可以是：link(网络设备)、addr(设备的协议地址)、route(路由表)、rule(策略)、neigh(arp 缓存)、tunnel(IP 通道)、maddr(多播地址)、mroute(多播路由)。

COMMAND 是操作命令，不同的对象有不同的命令配置。

link 对象支持的命令包括 set、show。

addr 对象支持的命令包括 add、del、flush、show。

route 对象支持的命令包括 list、flush、get、add、del、change、append、replace、monitor。

rule 对象支持的命令包括 list、add、del、flush。

neigh 对象支持的命令包括 add、del、change、replace、show、flush。

tunnel 对象支持的命令包括 add、change、del、show。

maddr 支持的命令包括 add、del。

mroute 支持的命令包括 show。

3.5.2　选项列表

详细参数说明如表 3-4 所示。

表 3-4　常用 ip 选项参数

选　　项	说　　明
-V | -Version	显示版本信息
--help	显示帮助信息
-s | -stats | -statistics	显示详细的信息
-f | -family	指定协议类型
-4	等同-family inet
-6	等同-family inet6
-0	等同-family link
-o | -oneline	每条记录输出一行
-r | -resove	使用系统名字解析 DNS

3.5.3　ip link(网络设备配置)

链路是一种网络设备,相应的命令可显示和改变设备的状态。

1. ip link set(改变设备属性)

devNAME(default),NAME 用于指定要操作的网络设备。配置 SR-IOV 虚拟功能(VF)设备时,此关键字应指定关联的物理功能(PF)设备。

up、down:改变设备的状态,即开或者关。

arp on、arp off:更改设备的 NOARP 标志。

multicast on、multicast off:更改设备的 MULTICAST 标志。

dynamic on、dynamic off:更改设备的 DYNAMIC 标志。

nameNAME:更改设备的名字,如果设备正在运行或者已经有一个配置好的地址,则操作无效。

txqueuelenNUMBER、txqlenNUMBER:更改设备发送队列的长度。

mtuNUMBER:更改设备 MTU。

addressLLADDRESS:更改接口的站点地址。

broadcastLLADDRESS、brdLLADDRESS、peerLLADDRESS:当接口为 POINTOPOINT

时，更改链路层广播地址或对等地址。

netnsPID：将设备移动到与进程 PID 关联的网络命名空间。

aliasNAME：给设备一个符号名以便于参考。

vfNUM：指定要配置的虚拟功能设备。必须使用 dev 参数指定关联的 PF 设备。

警告：如果请求更改多个参数，则在任何更改失败后立即中止 IP。这是 IP 能够将系统移动到不可预测状态的唯一情况。解决方案是避免使用一个 IP 链路集调用来更改多个参数。

2. ip link show(显示设备属性)

devNAME(default)：NAME 用于指定要显示的网络设备。如果省略此参数，则列出所有设备。

up：只显示运行的设备。

3.5.4 ip address(协议地址管理)

该地址是附加到网络设备上的协议(IP 或 IPv6)地址。每个设备必须至少有一个地址才能使用相应的协议。可以将几个不同的地址附加到一个设备上。这些地址不受歧视，因此别名一词不太适合它们。ip addr 命令用于显示地址及其属性，以及添加新地址并删除旧地址。

1. ip address add(增加新的协议地址)

devNAME：要向其添加地址的设备的名称。

localADDRESS(default)：接口的地址。地址的格式取决于协议。它是一个用于 IP 的虚线四边形和一系列十六进制半字，用冒号分隔，用于 IPv6。地址后面可以是斜杠和十进制数，它们用于编码网络前缀长度。

peerADDRESS：点对点接口的远程端点的地址。同样，地址后面可以是斜杠和十进制数，用于编码网络前缀长度。如果指定了对等地址，则本地地址不能具有前缀长度。网络前缀与对等端相关联，而不是与本地地址相关联。

broadcastADDRESS：接口的广播地址。可以使用特殊符号“+”和“-”代替广播地址。在这种情况下，通过设置/重置接口前缀的主机位来导出广播地址。

labelNAME：每个地址都可以用标签字符串标记。为了保持与 Linux 2.0 网络别名的兼容性，此字符串必须与设备名称重合，或者必须以设备名后跟冒号作为前缀。

scopeSCOPE_VALUE：地址有效的区域的范围。可用的作用域列在文件/etc/iproute2/rt_scopes 中。预定义的范围值如下。

(1) global：地址全局有效。

(2) site：(仅 IPv6)该地址为站点本地地址，即该地址在此站点内有效。

(3) link：该地址是本地链接，即它仅在此设备上有效。

(4) host：该地址仅在此主机内有效。

2. ip address delete(删除协议地址)

Arguments：与 ip addr add 的参数一致。设备名称是必需的参数。其余的参数都是可选的。如果没有提供参数，则删除第 1 个地址。

3. ip address show(显示协议地址)

devNAME(default)：设备名字。

scopeSCOPE_VAL：仅列出具有此作用域的地址。

toPREFIX：仅列出匹配 PREFIX 的地址。

labelPATTERN：只列出与模式匹配的标签的地址。

Dynamic、permanent：(仅 IPv6)仅列出由于无状态地址配置而安装的地址，或只列出永久(非动态)地址。

tentative：(仅 IPv6)仅列出未通过重复地址检测的地址。

deprecated：(仅 IPv6)列出废弃地址。

primary、secondary：只列出主(或辅助)地址。

4. ip address flush(刷新协议地址)

此命令用于刷新由某些条件选择的协议地址。此命令具有与 Show 相同的参数。不同之处在于，当不给出参数时，它不会运行。警告：这个命令(及下面描述的其他刷新命令)非常危险，因为会清除所有的地址。

使用-statistics 选项，命令变得详细。它会打印出已删除地址的数量和为刷新地址列表而进行的轮次数。如果提供了两次此选项，则 ip addr flush 也会以特定的格式转储所有已删除的地址。

3.5.5　ip addrlabel(协议地址标签管理)

IPv6 地址标签用于 RFC 3484 中描述的地址选择。优先级由用户空间管理，只有标签存储在内核中。

1. ip addrlabel add(增加地址标签)

prefixPREFIX、devDEV：输出接口。

labelNUMBER：prefix 的标签，0xffffffff 保留。

2. ip addrlabel del(删除地址标签)

该命令用于删除内核中的一个地址标签条目。参数：与 ip addrlabel add 的参数一致，但不需要标签。

3. ip addrlabel list(列出地址标签)

显示地址标签的内容。

4. ip addrlabel flush(刷新地址标签)

刷新地址标签的内容，并且不保存默认设置。

3.5.6　ip neighbour(邻居/ARP 表管理)

邻居对象为共享相同链路的主机建立协议地址和链路层地址之间的绑定。邻接条目被

组织成表。IPv4 邻居表的另一个名称是 ARP 表。相应的命令用于显示邻居绑定及其属性,以及添加新的邻居项并删除旧条目。

(1) ip neighbour add:增加邻居表。

(2) ip neighbour change:改变已经存在的邻居表。

(3) ip neighbour replace:增加一张表或者修改已经存在的表。

这些命令用于创建新的邻居记录或更新现有记录。上面的 3 个命令的使用方法如下。

toADDRESS(default):邻居的协议地址。它要么是 IPv4 地址,要么是 IPv6 地址。

devNAME:连接到邻居的接口。

lladdrLLADDRESS:邻居的链路层地址,可以是 null。

nudNUD_STATE:邻居的状态,可以是下面的值。

(1) permanent:邻居项永远有效,只能由管理员删除。

(2) noarp:邻居项有效。将不会尝试验证此条目,但可以在其生存期届满时删除该条目。

(3) reachable:邻居项在可达超时过期之前是有效的。

(4) stale:邻居的进入是有效的,但却是可疑的。如果邻居状态有效且此命令未更改地址,则此选项不会更改邻居状态。

1. ip neighbour delete(删除邻居表)

此命令使邻居项无效。这些参数与 ip neigh add 相同,只是将忽略 lladdr 和 nud。警告:试图删除或手动更改内核创建的 noarp 条目可能会导致不可预测的行为。特别是,即使在 NOARP 接口上,如果地址是多播或广播的,内核则可以尝试解析此地址。

2. ip neighbour show(显示邻居表)

toADDRESS(default):选择要列出的邻居的前缀。

devNAME:只列出与此设备相连的邻居。

unused:只列出当前未使用的邻居。

nudNUD_STATE:只列出此状态中的相邻项。NUD_STATE 接受下面列出的值或特殊值 all,这意味着接受所有状态。此选项可能发生不止一次。如果没有此选项,则 IP 会列出除 None 和 noarp 以外的所有条目。

3. ip neighbour flush(刷新邻居表)

此命令用于刷新相邻表,根据某些条件选择要刷新的条目。此命令具有与 show 相同的参数。不同之处在于,当不给出参数时,它不会运行,而要刷新的默认邻居状态不包括 permanent 和 noarp。

3.5.7 ip route(路由表管理)

操纵内核路由表中的路由条目并保存其他网络节点的路径信息。可能的路由类型如下。

(1) unicast:路由条目描述到路由前缀所涵盖的目的地的实际路径。

(2) unreachable：这些目的地是无法到达的。丢弃数据包，生成不可访问的 ICMP 消息主机。本地发件人得到一个 EHOSTUNEACH 错误。

(3) blackhole：这些目的地是无法到达的。数据包被静默丢弃。本地发送者得到一个 EINVAL 错误。

(4) prohibit：这些目的地是无法到达的。丢弃数据包并生成 ICMP 消息通信，该 ICMP 消息通信在管理上被禁止。本地发件人得到一个 EACCES 错误。

(5) local：将目的地分配给此主机。数据包被环回并在本地传送。

(6) broadcast：目的地是广播地址。数据包作为链路广播发送。

(7) throw：与策略规则一起使用的特殊控制路径。如果选择这样的路由，则将终止此表中的查找，假装没有找到路由。如果没有策略路由，则相当于路由表中没有路由。丢包并生成不可到达的 ICMP 消息网。本地发送者得到一个 ENETUNEACH 错误。

(8) nat：一条特殊的 NAT 路线。前缀覆盖的目的地被认为是虚拟地址(或外部地址)，需要在转发之前转换为真实地址(或内部地址)。选择要转换到的地址，并附带属性警告：Linux 2.6 中不再支持路由 NAT。

(9) via、anycast：未实现目标是分配给此主机的任意广播地址。它们主要等同于本地地址，但有一个不同之处：当将这些地址用作任何数据包的源地址时，这些地址是无效的。

multicast：用于多播路由的一种特殊类型。它不存在于普通路由表中。

路由表：Linux 2.x 可以将路由打包到从 1 到 255 的数字标识的多个路由表中，或者根据文件/etc/iucte 2/rt_tables 的名称，在默认情况下，所有普通路由都插入主表(ID 254)，内核只在计算路由时使用此表。实际上，另一张表总是存在的，这是不可见的，但更重要，即它是本地表(ID 255)。此表由本地地址和广播地址的路由组成。内核自动维护这个表，管理员通常不需要修改它，甚至不需要查看它。当使用策略路由时，多个路由表进入游戏。

(1) ip route add：增加路由。

(2) ip route change：修改路由。

(3) ip route replace：改变或者增加路由。

toTYPEPREFIX(default)：路由的目标前缀。如果省略类型，则 IP 采用类型单播。以上列出了其他类型的值。前缀是一个 IP 或 IPv6 地址，后面有斜杠和前缀长度。如果前缀的长度丢失，则 IP 将采用全长主机路由，还有一个特殊的前缀默认值，相当于 IP 0/0 或 to IPv6::/0。

tosTOS、dsfieldTOS：服务类型(TOS)密钥。该密钥没有关联的掩码，最长的匹配被理解为比较路由和数据包的 TOS。如果它们不相等，则数据包仍然可以匹配为 0TOS 的路由。TOS 要么是 8 位十六进制数字，要么是/etc/iproute2/rt_dsfield 中的标识符。

metricNUMBER、preferenceNUMBER：路由的首选值。NUMBER 是任意的 32 位数。

tableTABLEID：要添加此路由的表。TABLEID 可能是文件/etc/iproute2/rt_tables

中的一个数字或字符串。如果省略此参数，则 IP 假定主表，但本地路由、广播路由和 NAT 路由除外，在默认情况下这些路由被放入本地表中。

devNAME：输出设备名字。

viaADDRESS：下一个路由器的地址。实际上，这个字段的意义取决于路由类型。对于普通单播路由，它要么是真正的下一跳路由器，要么是安装在 BSD 兼容模式下的直接路由，它可以是接口的本地地址。对于 NAT 路由，它是已翻译的 IP 目的地块的第 1 个地址。

srcADDRESS：当发送到路由前缀所涵盖的目的地时要首选的源地址。

realmREALMID：指定此路由的域。REALMID 可能是文件/etc/iproute2/rt_realms 中的一个数字或字符串。

mtuMTU、mtulockMTU：沿着到达目的地的路径的 MTU。如果未使用修饰符锁，则由于路径 MTU 发现，内核可能更新 MTU。如果使用修饰符锁，则不会尝试路径 MTU 发现，所有数据包都将在 IPv4 情况下不使用 DF 位发送，或者在 IPv6 中碎片到 MTU。

windowNUMBER：TCP 向这些目的地进行广告的最大窗口，以字节为单位。它限制了 TCP 对等点允许发送给最大数据突发。

rttTIME：最初的 RTT（往返时间）估计。如果没有指定后缀，则单元会被直接传递给路由代码的原始值，以保持与以前版本的兼容性。否则如果使用 s、sec 或 secs 后缀指定秒，则使用 ms、msec 或 msecs 指定毫秒。

rttvarTIME(2.3.15+ only)：初始 RTT 方差估计。值与上述 RTT 指定的值相同。

rto_minTIME(2.6.23+ only)：与此目标通信时要使用的最小 TCP 重传超时。值与上述 RTT 指定的值相同。

ssthreshNUMBER(2.3.15+ only)：初始慢启动阈值的估计。

cwndNUMBER(2.3.15+ only)：阻塞窗口的夹子。如果不使用锁标志，则忽略它。

initcwndNUMBER：TCP 连接的 MSS 中的最大初始拥塞窗口（CWND）大小。

initrwndNUMBER(2.6.33+ only)：连接到此目标的初始接收窗口大小。实际窗口大小是此值乘以连接的 MSS。默认值为 0，意味着使用慢速开始值。

advmssNUMBER(2.3.15+ only)：MSS（最大段大小）在建立 TCP 连接时向这些目的地做广告。如果未给出，则 Linux 将使用从第 1 跳设备 MTU 中计算出来的默认值（如果到达这些目的地的路径是不对称的，则这种猜测可能是错误的）。

reorderingNUMBER(2.3.15+ only)：到达此目标的路径上的最大重排序。如果未给出，Linux 则将使用 sysctl 变量 net/ipv4/tcp_reordering 所选择的值。

nexthopNEXTHOP：多径路径的下一个。NEXTHOP 是一个复杂的值，其语法类似于顶级参数列表。

（1）viaADDRESS：下一个路由器。

（2）devNAME：输出设备

（3）weightNUMBER：是反映其相对带宽或质量的多径路由的此元素的权重。

scopeSCOPE_VAL：路由前缀所涵盖的目的地的范围。SCOPE_VAL 可以是文件/

etc/iproute2/rt_scopes 中的一个数字或字符串。如果省略此参数，则 IP 假定所有网关单播路由的作用域全局、直接单播和广播路由的范围链接及本地路由的范围主机。

protocolRTPROTO：此路由的路由协议标识符。RTPROTO 可以是文件/etc/iproute2/rt_protos 中的一个数字或字符串。如果未给出路由协议 ID，则 IP 假定协议启动（假定路由是由不了解他们正在做的事情的人添加的）。一些协议值有固定的解释。

（1）redirect：该路由是由于 icmp 重定向而安装的。

（2）kernel：该路由是由内核在自动配置期间安装的。

（3）boot：该路由是在启动过程中安装的。如果路由守护进程启动，则将清除所有守护进程。

（4）static：管理员安装了该路由以覆盖动态路由。路由守护进程将尊重它们，甚至可能会向其对等方发布广告。

（5）ra：路由是由路由器发现协议安装的。

Onlink：假装 Nextthop 直接连接到此链接，即使它不匹配任何接口前缀。

Equalize：允许在多径路由上逐包随机化。如果没有这个修饰符，路由则将被冻结到一个选定的下一个，这样负载拆分将只发生在每个流基上。只有当内核被修补时，均衡化才能工作。

ip route delete：删除路由。ip route del 与 ip route add 具有相同的参数，但它们的语义略有不同。键值（to、tos、首选项和表）选择要删除的路由。如果存在可选属性，则 IP 验证它们是否与要删除的路由的属性一致。如果没有找到具有给定密钥和属性的路由，则 ip route del 将失败。

ip route show：显示路由。toSELECTOR（default），仅从给定的目的地范围中选择路由。SELECTOR 由一个可选修饰符（root、match、exact）和一个前缀组成。root 用于选择前缀不小于 PREFIX 的路由。例如，root 0/0 选择整个路由表。match 用于选择前缀长度不超过 PREFIX 的路由。例如，match 10.0/16 选择 10.0/16、10/8 和 0/0，但未选择 10.1/16 和 10.0.0/24。exact（或仅仅前缀）用于选择具有此前缀的路由。如果这两个选项都没有出现，则 IP 假设为根 0/0，即它会列出整个表。

tosTOS，只选择具有给定 tos 的路由。

tableTABLEID 用于显示此表中的路线。默认设置为显示 tablemain。TABLEID 可以是实表的 ID，也可以是特殊值之一。

（1）all：列出所有的表。

（2）cache：列出路由缓存的内容。

cloned 或 cached：列出被克隆出来的路由，即由于某些路由属性改变，例如 MTU，而由某些路由派生出来的路由。（MTU）已更新。实际上，它等同于 table cache。

fromSELECTOR，语法与 to 相同，但它用于绑定源地址范围而不是目的地。需要注意，FROM 选项仅适用于克隆路由。

protocolRTPROTO：仅列出此路由的协议。

scopeSCOPE_VAL：仅列出具有此范围的路由。

typeTYPE：只列出此类型的路由。

devNAME：只列出经过此设备的路由。

viaPREFIX：只列出通过前缀选择的下一个路由器的路由。

srcPREFIX：只列出由前缀选择的首选源地址的路由。

realmREALMID 和 realmsFROMREALM/TOREALM：只列出这些领域的路由。

ip route flush：刷新路由表。此命令用于刷新由某些标准选择的路由，参数具有与 ip route show 的参数相同的语法和语义，但是路由表没有列出，而是被清除。唯一的区别是默认操作：显示转储所有 IP 主路由表，但刷新打印助手页。

使用-statistics 选项，命令变得详细。它打印出已删除路由的数目和刷新路由表的轮数。如果该选项被给予两次，则 IP 路由刷新也会以前面部分描述的格式转储所有已删除的路由。

ip route get：获取一个单独的路由，此命令用于获取一条到达目标的路由，并按照内核所看到的那样打印其内容。

toADDRESS(default)：目标地址。

fromADDRESS：源地址。

tosTOS、dsfieldTOS：服务类型。

iifNAME：预期将从该包到达的设备。

oifNAME：强制将此数据包路由的输出设备。

connected：如果没有提供源地址，则重新查找从第 1 次查找中接收的源设置并以此作为首选地址的路由。如果使用策略路由，则可能是不同的路由。

需要注意，此操作不等同于 ip route show。show 表示显示现有路线。如果必要，get 则可以解决它们并创建新的克隆。

3.5.8 ip rule(路由策略数据库管理)

rule 规则在路由策略数据库中控制路由选择算法。因特网中使用的经典路由算法只根据数据包的目的地地址(理论上，而不是实际中的 TOS 字段)进行路由决策。在某些情况下，希望通过不同的方式路由数据包，这不仅取决于目的地地址，还取决于其他数据包字段：源地址、IP、传输协议端口，甚至包有效负载。此任务称为“策略路由”。为了解决这一问题，传统的基于目标的路由表按照最长的匹配规则排序，被替换为“路由策略数据库”(RPDB)，该数据库通过执行一组规则来选择路由。

每个策略路由规则由一个选择器和一个动作谓词组成。RPDB 按照增加优先级的顺序进行扫描。每个规则的选择器应用于{源地址、目标地址、传入接口、tos、fwmark}，如果选择器与数据包匹配，则执行操作。动作谓词可能会成功返回。在这种情况下，它将给出路由或故障指示，并终止 RPDB 查找。否则 RPDB 程序将继续执行下一条规则。

语义上，自然动作是选择下一个和输出设备。在启动时，内核配置由三条规则组成的默

认 RPDB。

(1) Priority：0。Selector：匹配任何内容。Action：查找本地路由表(ID 255)。本地表是包含本地地址和广播地址的高优先级控制路由的特殊路由表。

(2) Priority：32766。Selector：匹配任何内容。Action：查找路由主表(ID 254)。主表是包含所有非策略路由的普通路由表。管理员可以删除和/或用其他规则重写此规则。

(3) Priority：32767。Selector：匹配任何内容。Action：查找路由表默认值(ID 253)。默认表为空。如果没有先前的默认规则选择数据包，则保留用于某些后处理。这一规则也可以删除。

RPDB 可能包含以下类型的规则。

(1) unicast：该规则规定返回在规则引用的路由表中找到的路由。

(2) blackhole：这条规则规定要悄悄丢弃数据包。

(3) unreachable：该规则规定生成“网络不可达”错误。

(4) prohibit：该规则规定产生“在行政上禁止通信”错误。

(5) nat：该规则规定将 IP 数据包的源地址转换为其他值。

ip rule add：增加规则。

ip rule delete：删除规则。

typeTYPE(default)：这个规则的类型。

fromPREFIX：选择要匹配的源前缀。

toPREFIX：选择要匹配的目标前缀。

iifNAME：选择要匹配的传入设备。如果接口是回送的，则该规则只匹配来自此主机的数据包。这意味着可以为转发包和本地数据包创建单独的路由表，从而完全隔离它们。

tosTOS、dsfieldTOS：选择要匹配的 TOS 值。

fwmarkMARK：选择要匹配的 fwmark 值。

priorityPREFERENCE：这条规则的优先级。每个规则都应该有一个显式设置的唯一优先级值。选项、偏好和顺序是优先级的同义词。

tableTABLEID：如果规则选择器匹配，则查找路由表标识符。还可以使用查找而不是表。

realmsFROM/TO：规则匹配和路由表查找成功时要选择的区域。只有当路由没有选择任何领域时，才使用要使用的领域。

natADDRESS：要翻译的 IP 地址块的基(用于源地址)。该地址可以是 NAT 地址块的开始(由 NAT 路由选择)，也可以是本地主机地址(甚至为 0)。在最后一种情况下，路由器不会翻译数据包，而是将它们伪装成这个地址。使用 map-to 而不是 NAT 意味着同样的事情。

ip rule flush：刷新规则，还转储所有已删除的规则。

ip rule show：显示规则，没有参数。

3.5.9 ip maddress(多播地址管理)

(1) ip maddress show：显示多播地址。devNAME(default)：设备名字。

(2) ip maddress add：增加多播地址。

(3) ip maddress delete：删除多播地址。

这些命令用于附加/分离一个静态链路层多播地址，以便在接口上侦听。需要注意，不可能静态地加入协议多播组。此命令仅管理链接层地址，主要参数如下。

addressLLADDRESS(default)：链路层多播地址。

devNAME：加入/删除多播地址的设备。

3.5.10 ip mroute(多播路由缓存管理)

mroute 对象是由用户级 mrouting 守护进程创建的多播路由缓存条目。由于组播路由引擎当前接口的局限性，无法对多播路由对象进行管理更改，因此只能显示对象。

ip mroute show：列出 mroute 缓存项。

toPREFIX(default)：选择要列出的目标多播地址的前缀。

iifNAME：接收多播数据包的接口。

fromPREFIX：选择多播路由的 IP 源地址的前缀。

3.5.11 ip tunnel(通道配置)

tunnel 对象是隧道，它将数据包封装在 IP 包中，然后通过 IP 基础结构发送。加密(或外部)地址族由-f 选项指定。默认的是 IPv4。

(1) ip tunnel add：增加一个新隧道。

(2) ip tunnel change：修改一个已经存在的隧道。

(3) ip tunnel delete：删除隧道。

nameNAME(default)：隧道设备名字。

modeMODE：设置隧道模式。可用的模式取决于封装地址系列。IPv4 封装可用的模式：ipip、SIT、isatap 和 grep；IPv6 封装的模式：ip6ip6、ipip6 和 any。

remoteADDRESS：设置隧道的远程端点。

localADDRESS：设置隧道数据包的固定本地地址。它必须是此主机的另一个接口上的地址。

ttlN：在隧道化的数据包上设置固定的 TTL N。N 是介于 1～255 的一个数字。0 是一个特殊值，意味着数据包继承 TTL 值。IPv 4 隧道的默认值为：Inherence。IPv6 隧道的默认值为 64。

tosT、dsfieldT、tclassT：在隧道数据包上设置固定的 TOS(或 IPv6 中的流量类)T。默认值为：inherit。

devNAME：将隧道绑定到设备名称，以便隧道数据包只能通过此设备路由，并且在到

达端点的路由发生更改时无法逃逸到另一个设备。

nopmtudisc：禁用此隧道上的路径 MTU 发现。在默认情况下启用它。需要注意，固定的 ttl 与此选项不兼容：使用固定的 ttl 进行隧道操作总会使 pmtu 发现。

keyK、ikey K、okey K：（只有 GRE 隧道）使用键控 GRE 与密钥 K，K 要么是一个数字要么是一个类似 IP 地址的虚线四边形。key 参数设置在两个方向上使用的键。ikey 和 okey 参数可为输入和输出设置不同的键。

csum、icsum、ocsum：（只有 GRE 隧道）生成/要求隧道数据包的校验和。ocsum 标志计算传出数据包的校验和。icsum 标志要求所有输入数据包都具有正确的校验和。csum 标志等效于组合 icsum ocsum。

seq、iseq、oseq：（只有 GRE 隧道）序列化数据包。oseq 标志允许对传出数据包进行排序。iseq 标志要求对所有输入数据包进行序列化。seq 标志等效于 iseq oseq 组合。这不是工作，不要用它。

dscpinherit：（只有 IPv6 隧道）在内部和外部报头之间继承 DS 字段。

encaplimELIM：设置固定的封装限制。默认值为 4。

flowlabelFLOWLABEL：（只有 IPv6 隧道）设置固定的流标签。

（4）ip tunnel prl：潜在路由器列表（只有 ISATAP），主要参数：devNAME、prl-defaultADDR、prl-nodefaultADDR、prl-deleteADDR，添加或删除 addr 作为潜在的路由器或默认路由器。

（5）ip tunnel show：列出隧道，没有参数，直接查看所有隧道信息。

3.5.12　ip monitor and rtmon（状态监控）

IP 实用程序可以连续地监视设备、地址和路由的状态。此选项的格式略有不同。也就是说，监视器 command 是命令行中的第 1 个命令，对象列表如下：

```
ip monitor [ all | LISTofOBJECTS]
```

OBJECT-LIST 是要监视的对象类型的列表。它可能包含链接、地址和路由。如果没有提供文件参数，则 IP 将打开 RTNETLINK，侦听该参数，并以前面部分描述的格式转储状态更改。

如果给定文件名，则不会侦听 RTNETLINK，而是打开包含以二进制格式保存的 RTNETLINK 消息的文件，并将其转储。可以使用 rtmon 实用程序生成这样的历史文件。此实用程序具有类似于 IP 监视器的命令行语法。在理想情况下，应该在发出第 1 个网络配置命令之前启动 rtmon。例如，在一个启动脚本中插入：

```
rtmon file /var/log/rtmon.log
```

稍后将能够查看完整的历史记录。当然，可以随时启动 rtmon。它会在历史记录的前面加上在启动时转储的状态快照。

3.5.13 ip xfrm(设置 xfrm)

xfrm 是一个 IP 框架,它可以转换数据报文的格式,即用某种算法对数据包进行加密。xfrm 策略和 xfrm 状态通过模板 tmpl_list 相关联。该框架被用作 IPSec 协议的一部分。

(1) ip xfrm state add:增加新的状态。

(2) ip xfrm state update:更新已经存在的状态。

(3) ip xfrm state allocspi:分配 SPI 数值。

MODE:设置为默认传输,但可以设置为 tunnel、ro 或者 beet。

FLAG-LIST:包含一个或多个标志。

FLAG:可以设置为 noecn、decap-dscp、wildrecv。

ENCAP:封装设置为封装类型 ENCAP-TYPE、源端口 SPORT、目标端口 DPORT 和 OADDR。

ENCAP-TYPE:可以是 espinudp 或者 espinudp-nonike。

ALGO-LIST:包含一个或多个算法 Algo,该算法依赖于 Algo_type 设置的算法类型。它可以使用算法 enc、auth、comp。

(4) ip xfrm policy add:增加新策略。

(5) ip xfrm policy update:更新已经存在的策略。

(6) ip xfrm policy delete:删除存在的策略。

(7) ip xfrm policy get:过去存在的策略。

(8) ip xfrm policy deleteall:删除所有的 xfrm 策略。

(9) ip xfrm policy list:打印策略列表。

(10) ip xfrm policy flush:刷新策略。

dirDIR:目录可以是 inp、out、fwd。

SELECTOR:选择将设置策略的地址。选择器由源地址和目标地址定义。

UPSPEC:由源端口 sport、目的端口 dport、type 和 code 定义。

devDEV:指定网络设备。

indexINDEX:索引策略的数量。

ptypePTYPE:默认为 main,可以切换为 sub。

actionACTION:默认为 allow,可以切换为 block。

priorityPRIORITY:级别是一个数字,默认为 0。

LIMIT-LIST:限制以秒、字节或数据包数量为单位进行设置。

TMPL-LIST:模板列表基于 ID、mode、reqid、level。

ID:由源地址、目标地址、proto 和 spi 的值指定。

XFRM_PROTO:值可以是 esp、ah、comp、route2、hao。

MODE:默认为 transport,还可以是 tunnel、beet。

LEVEL:默认为 required,还可以是 use。

UPSPEC：由 sport、dport、type、code 指定。

(11) ip xfrm monitor：用于列出所有对象或定义的对象组。xfrm monitor 可以监视所有对象或其中定义的组的策略。

3.5.14　ip token

IPv6 令牌化接口标识支持用于向节点分配众所周知的主机部分地址，同时仍然从路由器广告获得全局网络前缀。令牌标识符的主要目标是服务器平台，其中的地址通常是手动配置的，而不是使用 DHCPv6 或 SLAAC。通过令牌化标识符，主机仍然可以使用 SLAAC 确定其网络前缀，但如果其网络前缀被更改，则更容易自动重新编号[1]。

(1) ip token set：设置接口令牌。TOKEN 为接口标识符令牌地址；devDEV 为网络接口。

(2) ip token get：从内核获取接口令牌，显示特定网络设备的令牌化接口标识符。参数与 ip token set 的参数一致，但必须省略该令牌。

(3) ip token list：列出所有接口令牌，列出内核中网络接口的所有令牌化接口标识符。

3.5.15　简要实例说明

1. 显示设备的各种协议地址

显示设备支持的协议的地址，命令如下：

```
[root@localhost ~]# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 08:00:27:14:33:57 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.9/24 brd 192.168.1.255 scope global eth0
    inet6 fe80::a00:27ff:fe14:3357/64 scope link
       valid_lft forever preferred_lft forever
```

2. 为目标设备添加地址

查看帮助文档，命令如下：

```
[root@localhost ~]# ip addr help
Usage: ip addr {add|change|replace} IFADDR dev STRING [ LIFETIME ]
                                                      [ CONFFLAG-LIST]
       ip addr del IFADDR dev STRING
       ip addr {show|flush} [ dev STRING ] [ scope SCOPE-ID ]
                            [ to PREFIX ] [ FLAG-LIST ] [ label PATTERN ]
```

给 eth0 添加新的 IP，命令如下：

```
[root@localhost ~]# ip addr add 192.168.1.110 dev eth0
```

查看 eth0 的地址信息，多了一个 IP，命令如下：

```
[root@localhost ~]# ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 08:00:27:14:33:57 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.9/24 brd 192.168.1.255 scope global eth0
    inet 192.168.1.110/32 scope global eth0
    inet6 fe80::a00:27ff:fe14:3357/64 scope link
       valid_lft forever preferred_lft forever
```

3.5.16 传统网络配置命令与 ip 高级路由命令

传统的网络配置 ifconfig 在 1～3 点，ip 高级路由命令在 4～12 点，两者部分可以通用，并达到同样的目的，但 ip 命令的功能更强大，可以实现更多的配置目的。

1. 使用 ifconfig 命令配置并查看网络接口情况

配置 eth0 的 IP，同时激活设备，命令如下：

```
# ifconfig eth0 192.168.4.1 netmask 255.255.255.0 up
```

配置 eth0 别名设备 eth0:1 的 IP，并添加路由，命令如下：

```
# ifconfig eth0:1 192.168.4.2
# route add -host 192.168.4.2 dev eth0:1
```

激活(禁用)设备，命令如下：

```
# ifconfig eth0:1 up(down)
```

查看所有(指定)网络接口配置，命令如下：

```
# ifconfig (eth0)
```

2. 使用 route 命令配置路由表

添加到主机路由，命令如下：

```
# route add -host 192.168.4.2 dev eth0:1
# route add -host 192.168.4.1 gw 192.168.4.250
```

添加到网络的路由，命令如下：

```
# route add -net IP netmask MASK eth0
# route add -net IP netmask MASK gw IP
# route add -net IP/24 eth1
```

添加默认网关，命令如下：

```
#route add default gw IP
```

删除路由，命令如下：

```
#route del -host 192.168.4.1 dev eth0:1
```

查看路由信息，命令如下：

```
#route 或 route -n（-n 表示不解析名字，列出速度会比 route 快）
```

3. ARP 管理命令

查看 ARP 缓存，命令如下：

```
#arp
```

添加 ARP 映射关系，命令如下：

```
#arp -s IP MAC
```

删除 ARP 映射关系，命令如下：

```
#arp -d IP
```

ip 是 iproute2 软件包里面的一个强大的网络配置工具，它能够替代一些传统的网络管理工具。例如 ifconfig、route 等，上面的示例完全可以用下面的 ip 命令实现，而且 ip 命令可以实现更多的功能。下面是一些示例。

ip 命令的语法，命令如下：

```
ip [OPTIONS] OBJECT [COMMAND [ARGUMENTS]]
```

（1）ip link set 用于改变设备的属性，缩写为 set、s。

up/down 起动/关闭设备，命令如下：

```
#ip link set dev eth0 up
```

这个等于传统的 ifconfig 用法，命令如下：

```
#ifconfig eth0 up(down)
```

改变设备传输队列的长度。

参数为 txqueuelen NUMBER 或者 txqlen NUMBER，命令如下：

```
#ip link set dev eth0 txqueuelen 100
```

改变网络设备 MTU(最大传输单元)的值,命令如下:

```
# ip link set dev eth0 mtu 1500
```

修改网络设备的 MAC 地址,参数为 address LLADDRESS,命令如下:

```
# ip link set dev eth0 address 00:01:4f:00:15:f1
```

(2) ip link show:显示设备属性,缩写为 show、list、lst、sh、ls、l。

-s 选项出现两次或者更多次,ip 命令会输出更为详细的错误信息统计,命令如下:

```
# ip -s -s link ls eth0
    eth0: mtu 1500 qdisc cbq qlen 100
    link/ether 00:a0:cc:66:18:78 brd ff:ff:ff:ff:ff:ff
    RX: Bytes packets errors dropped overrun mcast
    2449949362 2786187 0 0 0 0
    RX errors: length crc frame fifo missed
       0 0 0 0 0
    TX: Bytes packets errors dropped carrier collsns
    178558497 1783946 332 0 332 35172
    TX errors: aborted fifo window heartbeat
       0 0 0 332
```

这个命令等于传统的 ifconfig,命令如下:

```
ifconfig eth0
```

(3) ip address add:添加一个新的协议地址,缩写为 add、a。

为每个地址设置一个字符串作为标签。为了和 Linux 2.0 的网络别名兼容,这个字符串必须以设备名开头,接着一个冒号,命令如下:

```
# ip addr add local 192.168.4.1/28 brd + label eth0:1 dev eth0
```

在以太网接口 eth0 上增加一个地址 192.168.20.0,掩码长度为 24 位(155.155.155.0),标准广播地址,标签为 eth0:Alias,命令如下:

```
# ip addr add 192.168.4.2/24 brd + dev eth1 label eth1:1
```

这个命令等于传统的 ifconfig,命令如下:

```
ifconfig eth1:1 192.168.4.2
```

(4) ip address delete:删除一个协议地址,缩写为 delete、del、d。命令如下:

```
# ip addr del 192.168.4.1/24 brd + dev eth0 label eth0:Alias1
```

(5) ip address show:显示协议地址,缩写为 show、list、lst、sh、ls、l。命令如下:

```
# ip addr ls eth0
```

(6) ip address flush：清除协议地址，缩写为 flush、f。

删除属于私网 10.0.0.0/8 的所有地址，命令如下：

```
# ip -s -s a f to 10/8
```

取消所有以太网卡的 IP 地址，命令如下：

```
# ip -4 addr flush label "eth0"
```

(7) ip neighbour：neighbour/arp 表管理命令，缩写为 neighbour、neighbor、neigh、n。命令选项有 add、change、replace、delete、flush、show(或者 list)。

ip neighbour add：添加一个新的邻接条目。

ip neighbour change：修改一个现有的条目。

ip neighbour replace：替换一个已有的条目。

缩写分别为 add、a；change、chg；replace、repl。

在设备 eth0 上，为地址 10.0.0.3 添加一个 permanent ARP 条目，命令如下：

```
# ip neigh add 10.0.0.3 lladdr 0:0:0:0:0:1 dev eth0 nud perm
```

把状态改为 reachable，命令如下：

```
# ip neigh chg 10.0.0.3 dev eth0 nud reachable
```

ip neighbour delete：删除一个邻接条目。

删除设备 eth0 上的一个 ARP 条目 10.0.0.3，命令如下：

```
# ip neigh del 10.0.0.3 dev eth0
```

ip neighbour show：显示网络邻居的信息，缩写为 show、list、sh、ls。

显示某个 IP 的网络邻居信息，命令如下：

```
# ip -s n ls 193.233.7.254
193.233.7.254. dev eth0 lladdr 00:00:0c:76:3f:85 ref 5 used 12/13/20 nud reachable
```

ip neighbour flush：清除邻接条目，缩写为 flush、f。

-s 可以显示详细信息，命令如下：

```
# ip -s -s n f 193.233.7.254
```

(8) 路由表管理，缩写为 route、ro、r。从 Linux 2.2 开始，内核把路由归纳到许多路由表中，这些表都进行了编号，编号数字的范围是 1～255。另外，为了方便，还可以在/etc/

iproute2/rt_tables 中为路由表命名。在默认情况下，所有的路由都会被插入表 main(编号 254)中。在进行路由查询时，内核只使用路由表 main。

ip route add：添加新路由。

ip route change：修改路由。

ip route replace：替换已有的路由。

缩写分别为 add、a；change、chg；replace、repl。

设置到网络 10.0.0/24 的路由经过网关 193.233.7.65，命令如下：

```
# ip route add 10.0.0/24 via 193.233.7.65
```

修改到网络 10.0.0/24 的直接路由，使其经过设备 dummy，命令如下：

```
# ip route chg 10.0.0/24 dev dummy
```

实现链路负载平衡，加入缺省多路径路由，让 ppp0 和 ppp1 分担负载，scope 值并非必须，它只不过是告诉内核，这个路由要经过网关而不是直连的。实际上，如果知道远程端点的地址，则使用 via 参数设置就更好了，命令如下：

```
# ip route add default scope global nexthop dev ppp0 nexthop dev ppp1
# ip route replace default scope global nexthop dev ppp0 nexthop dev ppp1
```

设置 NAT 路由。在转发来自 192.203.80.144 的数据包之前，先进行网络地址转换，把这个地址转换为 193.233.7.83，命令如下：

```
# ip route add nat 192.203.80.142 via 193.233.7.83
```

实现数据包级负载平衡，允许把数据包随机从多个路由发出。weight 可以设置权重，命令如下：

```
# ip route replace default equalize nexthop via 211.139.218.145 dev eth0 weight 1 nexthop via
211.139.218.145 dev eth1 weight 1
```

ip route delete：删除路由，缩写为 delete、del、d。

删除已加入的多路径路由，命令如下：

```
# ip route del default scope global nexthop dev ppp0 nexthop dev ppp1
```

ip route show：列出路由，缩写为 show、list、sh、ls、l。

计算使用 GATED/BGP 协议的路由个数，命令如下：

```
# ip route ls proto gated/bgp |wc
1413 9891 79010
```

计算路由缓存里面的条数，由于被缓存路由的属性可能大于一行，所以需要使用-o 选

项，命令如下：

```
#ip -o route ls cloned |wc
159 2543 18707
```

列出路由表 TABLEID 里面的路由。缺省设置是 table main。TABLEID 或者一个真正的路由表 ID 或者/etc/iproute2/rt_tables 文件定义的字符串，或者以下的特殊值。

all：列出所有表的路由。

cache：列出路由缓存的内容。

查询 cache 表中的路由，命令如下：

```
ip ro ls 193.233.7.82 tab cache
```

列出某个路由表的内容，命令如下：

```
#ip route ls table fddi153
```

列出默认路由表的内容，命令如下：

```
#ip route ls
```

这个命令等于传统的 route。

ip route flush：擦除路由表。

删除路由表 main 中的所有网关路由，命令如下：

```
#ip -4 ro flush scope global type unicast
```

清除所有被克隆出来的 IPv6 路由，命令如下：

```
#ip -6 -s -s ro flush cache
```

在 gated 程序挂掉之后，清除所有的 BGP 路由，命令如下：

```
#ip -s ro f proto gated/bgp
```

清除所有 IPv4 路由 cache，命令如下：

```
#ip route flush cache
*** IPv4 routing cache is flushed.
```

ip route get：获得单个路由，缩写为 get、g

使用这个命令可以获得到达目的地址的一个路由及它的确切内容。

ip route get 命令和 ip route show 命令执行的操作是不同的。ip route show 命令只显示现有的路由，而 ip route get 命令在必要时会派生出新的路由。

搜索到 193.233.7.82 的路由，命令如下：

```
#ip route get 193.233.7.82
193.233.7.82 dev eth0 src 193.233.7.65 realms inr.ac cache mtu 1500 rtt 300
```

搜索目的地址是 193.233.7.82，来自 193.233.7.82，从 eth0 设备到达的路由，这条命令会产生一条非常有意思的路由，这是一条到 193.233.7.82 的回环路由，命令如下：

```
#ip r g 193.233.7.82 from 193.233.7.82 iif eth0
193.233.7.82 from 193.233.7.82 dev eth0 src 193.233.7.65 realms inr.ac/inr.ac
cache ; mtu 1500 rtt 300 iif eth0
```

(9) ip route：路由策略数据库管理命令。命令选项有 add、delete、show(或者 list)。策略路由(Policy Routing)不等于路由策略(Routing Policy)。

在某些情况下，不仅需要通过数据包的目的地址决定路由，可能还需要通过其他一些域：源地址、IP、传输层端口甚至数据包的负载。这就叫作策略路由。

ip rule add：插入新的规则。

ip rule delete：删除规则。

缩写分别为 add、a；delete、del、d。

通过路由表 inr.ruhep 路由来自源地址为 192.203.80/24 的数据包，命令如下：

```
ip ru add from 192.203.80/24 table inr.ruhep prio 220
```

把源地址为 193.233.7.83 的数据报文的源地址转换为 192.203.80.144，并通过表 1 进行路由，命令如下：

```
ip ru add from 193.233.7.83 nat 192.203.80.144 table 1 prio 320
```

删除无用的缺省规则，命令如下：

```
ip ru del prio 32767
```

ip rule show：列出路由规则，缩写为 show、list、sh、ls、l。

查看路由规则，命令如下：

```
#ip ru ls
0: from all lookup local
32762: from 192.168.4.89 lookup fddi153
32764: from 192.168.4.88 lookup fddi153
32766: from all lookup main
32767: from all lookup 253
```

(10) ip maddress：多播地址管理，缩写为 show、list、sh、ls、l。

ip maddress show：列出多播地址。

ip maddress add：加入多播地址。

ip maddress delete：删除多播地址。

后两条命令的缩写分别为 add、a；delete、del、d。

使用这两个命令，可以添加/删除在网络接口上监听的链路层多播地址。这两个命令只能管理链路层地址。

增加多播地址，命令如下：

```
#ip maddr add 33:33:00:00:00:01 dev dummy
```

查看多播地址，命令如下：

```
#ip -O maddr ls dummy
link 33:33:00:00:00:01 users 2 static
link 01:00:5e:00:00:01
```

删除多播地址，命令如下：

```
#ip maddr del 33:33:00:00:00:01 dev dummy
```

(11) ip mroute：多播路由缓存管理。ip mroute show：列出多播路由缓存条目，缩写为 show、list、sh、ls、l。

查看多播路由，命令如下：

```
#ip mroute ls
(193.232.127.6, 224.0.1.39) Iif: unresolved
(193.232.244.34, 224.0.1.40) Iif: unresolved
(193.233.7.65, 224.66.66.66) Iif: eth0 Oifs: pimreg
```

查看多播路由，命令如下：

```
#ip -s mr ls 224.66/16
(193.233.7.65, 224.66.66.66) Iif: eth0 Oifs: pimreg
9383 packets, 300256 Bytes
```

(12) ip tunnel：通道配置，缩写为 tunnel、tunl。

ip tunnel add：添加新的通道。

ip tunnel change：修改现有的通道。

ip tunnel delete：删除一个通道。

缩写分别为 add、a；change、chg；delete、del、d。

建立一个点对点通道，最大 TTL 是 32，命令如下：

```
#ip tunnel add Cisco mode sit remote 192.31.7.104 local 192.203.80.1 ttl 32
```

ip tunnel show：列出现有的通道，缩写为 show、list、sh、ls、l。

列出现有的通道，命令如下：

```
# ip -s tunl ls Cisco
```

(13) ip monitor 和 rtmon：状态监视，ip 命令可以用于连续地监视设备、地址和路由的状态。这个命令选项的格式有点不同，命令选项的名字叫作 monitor，接着是操作对象，命令格式如下：

```
ip monitor [ file FILE ] [ all | OBJECT-LIST ]
```

rtmon 状态监视，命令如下：

```
# rtmon file /var/log/rtmon.log
```

ip monitor 状态监视，命令如下：

```
# ip monitor file /var/log/rtmon.log r
```

4. 查看网络接口信息

要查看网络接口信息，例如 IP 地址、子网等，可使用 ip addr show 命令，命令如下：

```
# ip addr show
```

这会显示系统上所有网络接口的信息，但是如果要查看单个网卡信息，则可以使用以下命令查看 ens33 接口的 IP 信息，命令如下：

```
# ip addr show ens33
```

启用或者禁用网络接口可以使用 ip 命令，命令如下：

```
# sudo ip link set ens33 down
```

可以看到 ens33 接口的状态变成 DOWN 了。

再启用该网络接口，命令如下：

```
# sudo ip link set ens33 up
```

为接口设置临时的 IP 地址，要分配 IP 地址以使用 ip 命令进行接口，命令如下：

```
# sudo ip addr add 192.168.43.175/255.255.255.0 dev ens33
```

可以看到 ens33 接口添加了一个新的 IP 地址。

从网络接口中删除 IP 地址，如果要从接口中删除已分配的 IP，则命令如下：

```
# sudo ip addr del 192.168.43.175/24 dev ens33
```

查看路由信息会显示数据包到达目的地所要经过的路由。要检查网络路由信息，命令如下：

```
bob@Ubuntu-20-04:~$ ip route show
default via 192.168.43.2 dev ens33 proto dhcp metric 100
169.254.0.0/16 dev ens33 scope link metric 1000
192.168.43.0/24 dev ens33 proto Kernel scope link src 192.168.43.174 metric 100
```

在上面的输出中，可以看到所有网络接口的路由信息。还可以使用以下方式获取特定IP的路由信息，命令如下：

```
#ip route get to 192.168.43.2
```

查看ARP条目，ARP是“地址解析协议”的缩写，用于将IP地址转换为MAC地址，并且所有IP及其对应的MAC详细信息都存储在被称为ARP缓存的表中。要查看ARP缓存中的条目可以使用的命令如下：

```
#ip neigh
```

查看网络统计，使用ip命令可以查看所有网络接口的网络统计信息，例如传输的字节和数据包，以及错误或丢失的数据包等。要查看网络统计信息，命令如下：

```
#ip -s link
```

5. 条件路由策略匹配

(1) fwmark：将fwmark作为匹配条件时，必须搭配Netfilter一起使用，这看起来很麻烦，却是最灵活的匹配条件。例如，某公司对外有3条ADSL，希望所有HTTP经由第1条ADSL，SMTP及POP3经由第2条ADSL，其余流量则经由第3条ADSL。可以使用如下的命令组合来达到这样的目的：

```
iptables -t mangle -A FORWARD -i eth3 -p tcp --dport 80 -j MARK --set-mark 1
iptables -t mangle -A FORWARD -i eth3 -p tcp --dport 25 -j MARK --set-mark 2
iptables -t mangle -A FORWARD -i eth3 -p tcp --dport 110 -j MARK --set-mark 2
iptables -t mangle -A FORWARD -i eth3 -j MARK --set-mark 3
ip rule add fwmark 1 table 1
ip rule add fwmark 2 table 2
ip rule add fwmark 3 table 3
```

首先使用Netfilter的mangle机制针对特定的数据包设置MARK值，在此将HTTP数据包的MARK值设置为1，将SMTP及POP3数据包的MARK值设置为2，其余数据包则将MARK值设置为3。接着，再根据fwmark条件来判断数据包的MARK值，如果MARK值为1，则参考路由表1将数据包送出；如果MARK值为2，则参考路由表2将数据包送出；最后，如果MARK值为3，则参考路由表3将数据包送出。

以上示例只是一个概念而已，如果真要完整地体现出这个示例的所有功能，则需要注意

许多细节，稍后将使用详细的示例讲解这部分内容，在此只要了解 fwmark 与 Netfilter 结合使用的概念即可。

（2）dev：使用数据包输入的接口来作为判断依据，例如，希望凡是由 eth2 接口送入的数据包都由 eth0 接口转发出去，由 eth3 接口送入的数据包都由 eth1 接口转发出去。通过以下命令组合将能满足要求：

```
ip rule add dev eth2 table 1
ip rule add dev eth3 table 3
```

（3）优先级别：前面介绍了规则中“条件”的使用方式，接下来要讨论的是优先级别。优先级别用数字来表示，其范围可为 0～4 亿，堪称天文数字，实际上不可能在一台 PC 上设置如此庞大的路由机制。增加部分路由策略，命令如下：

```
[root@localhost ~]# ip rule show
0: from all lookup local
32766: from all lookup main
32767: from all lookup default
[root@localhost ~]#
[root@localhost ~]# ip rule add from 192.168.1.0/24 table 1
[root@localhost ~]# ip rule add from 192.168.2.0/24 table 2
[root@localhost ~]#
[root@localhost ~]# ip rule show
0: from all lookup local
32764: from 192.168.2.0/24 lookup 2
32765: from 192.168.1.0/24 lookup 1
32766: from all lookup main
32767: from all lookup default
```

如以上示例，执行 ip rule show 命令后所显示内容的第 1 个字段就是优先级别，数字越小，代表优先级别越高，也代表这条规则可以排得越靠前，如此数据包在进行条件匹配时，就会越早匹配到这条规则，从输出的数据中可知，默认优先级别 0、32766 及 32767 已被占用，因此，在添加规则时，如果没有特别设置优先级别，则优先级别默认会从 32766 开始递减，如 32765、32764……，如果需要特别设置优先级别，则可以在 ip rule add 命令的最后加上 prio XXX 参数，命令如下：

```
[root@localhost ~]# ip rule show
0: from all lookup local
32766: from all lookup main
32767: from all lookup default
[root@localhost ~]#
[root@localhost ~]# ip rule add from 192.168.1.0/24 table 1 prio 10
[root@localhost ~]# ip rule add from 192.168.2.0/24 table 2 prio 20
[root@localhost ~]#
[root@localhost ~]# ip rule show
0: from all lookup local
10: from 192.168.1.0/24 lookup 1
```

```
20: from 192.168.2.0/24 lookup 2
32766: from all lookup main
32767: from all lookup default
```

在 Linux 的基于策略的路由中，路由表用 ID 来表示，但如有必要，还可以用 ID 与名称对照表将 ID 转换成名称。

（4）删除规则：ip 命令提供的删除规则的方式十分灵活。例如，要删除下列第 2 条规则，可以分别使用“优先级别”“条件”及“路由表”当中任何一个唯一的值设置所需删除的规则，命令如下：

```
ip rule del prio 10
ip rule del from 192.168.1.0/24
ip rule del table 1
ip rule del from 192.168.1.0/24 table 1 prio 10

[root@localhost ~]# ip rule show
0: from all lookup local
10: from 192.168.1.0/24 lookup 1
20: from 192.168.2.0/24 lookup 2
32766: from all lookup main
32767: from all lookup default
[root@localhost ~]#
```

6. 路由表管理

由于 route -n 命令已经完全不适合在基于策略的路由中使用，因此，route 命令仅能操作一个特定的路由表，但在基于策略的路由中，会同时存在多个路由表，应放弃这个路由管理工具，取而代之的依然是 ip 命令。接下来将讨论如何使用 ip 命令来管理路由表。

（1）查看路由表内容，在查看路由表之前，首先使用 ip rule show 命令来查看目前使用了哪些路由表，接着，使用 ip route show [table id | name]命令来查看路由表的内容。例如，可以使用 ip route show table main 命令来查看路由表 main 的内容，如果省略路由表名称（如 ip route show），则会默认查看路由表 main 的内容。

```
[root@localhost /]# ip rule show
0: from all lookup local
32766: from all lookup main
32767: from all lookup default
[root@localhost /]#
[root@localhost /]# ip route show table main
10.10.15.0/25 dev eth0 proto Kernel scope link src 10.10.15.46
192.168.1.0/24 dev eth1 proto Kernel scope link src 192.168.1.10
default via 10.10.15.1 dev eth0
[root@localhost /]#
```

在默认情况下，系统有 3 个路由表，这 3 个路由表的功能如下。

local：路由表 local 包含本机路由及广播信息。例如，在本机上执行 ssh 127.0.0.1 时，

就会参考这份路由表的内容，在正常情况下，只要配置好网卡的网络设置，就会自动生成local路由表的内容，也不必修改其内容。

main：使用传统命令 route -n 所看到的路由表就是 main 的内容。Linux 系统在默认情况下使用这份路由表的内容来传输数据包，因此，其内容极为重要，在正常情况下，只要配置好网卡的网络设置，就会自动生成 main 路由表的内容。

default：最后是 default 路由表，这个路由表在默认情况下内容为空；除非有特别的要求，否则保持其内容为空即可。

在此使用路由表 main 的内容进行解释，因为在主机上有 eth0 及 eth1 两块网卡，并且为其设置的 IP 分别是 10.10.15.46/25 及 192.168.1.10/24，因此，路由表内的第 1 行即是告诉系统，如果有数据包要送到 10.10.15.0/25 这个网段，就直接将数据包由 eth0 接口送出，而本机临近这个网段的 IP 是 10.10.15.46，第 2 行则是设置到 192.168.1.0/24 的路由，其含义与第 1 行完全相同；以上这两行是只要将计算机网卡上的 IP 设置好，并在网络服务重启之后，默认就会生成的路由，无须特别设置。最后一行则指：如果数据包不是送往 10.10.15.0/25 及 192.168.1.0/24 网段，则数据包将统一转发给 10.10.15.1 主机去处理，而 10.10.15.1 就是在网络配置中所设置的“默认网关”。

```
[root@localhost /]# ip route show table main
10.10.15.0/25 dev eth0 proto Kernel scope link src 10.10.15.46
192.168.1.0/24 dev eth1 proto Kernel scope link src 192.168.1.10
default via 10.10.15.1 dev eth0
[root@localhost /]#
```

(2) 添加路由：添加路由在此还是采用 ip 命令而不是 route 命令，下例首先使用 ip route show 命令显示路由表 main 的内容，接着使用 ip route add 命令将所需的路由添加到路由表 main 中。最后使用 ip route show 命令将路由表 main 的内容打印出来，此时就可以在路由表 main 中看到刚才添加的路由了。完整的命令如下：

```
[root@localhost /]# ip route show table main
10.10.15.0/25 dev eth0 proto Kernel scope link src 10.10.15.46
192.168.1.0/24 dev eth1 proto Kernel scope link src 192.168.1.10
default via 10.10.15.1 dev eth0
[root@localhost /]#
[root@localhost /]# ip route add 192.168.2.0/24 via 10.10.15.50 table main
[root@localhost /]#
[root@localhost /]# ip route show table main
10.10.15.0/25 dev eth0 proto Kernel scope link src 10.10.15.46
192.168.2.0/24 via 10.10.15.50 dev eth0
192.168.1.0/24 dev eth1 proto Kernel scope link src 192.168.1.10
default via 10.10.15.1 dev eth0
[root@localhost /]#
```

如果要添加的路由并未出现在现有的路由表中，则该如何处理呢？在此有一个概念，单纯添加路由表并无意义，因为新增的路由表，系统默认为不会去使用，如果要将路由添加到

main 以外的路由表，则只有先添加“规则”才能确定新的路由表名称（Table ID），有了新的路由表之后，才会把路由添加到新的路由表中。

使用以下示例来说明这个过程。首先使用 ip rule show 命令来查询 RPDB 的当前状态，可以看到目前只有 3 条默认规则，接着，使用 ip rule add 命令来添加一条规则，此时系统内就多了一个有用的路由表，其路由表 ID 为 10，可以立即使用 ip route show 命令来查看这个新的路由表，其内容默认为空，接着可以在这个新路由表中添加路由，在此使用 ip route add 命令来添加路由，决定凡是来自 192.168.2.0/24 网段的数据包都从 eth1 接口将数据包送离本机，因此，必须完整编写 eth1 接口的路由。首先将临近 eth1 接口的路由填入，告诉系统本机与 192.168.1.0/24 网段的通信都通过 eth1 接口来处理，接着填入这个路由表的默认路由，最后使用 ip route show 命令显示路由表 10 的内容。完整的命令如下：

```
[root@localhost ~]# ip rule show
0: from all lookup local
32766: from all lookup main
32767: from all lookup default
[root@localhost ~]#
[root@localhost ~]# ip rule add from 192.168.2.0/24 table 10
[root@localhost ~]#
[root@localhost ~]# ip route show table 10
[root@localhost ~]#
[root@localhost ~]# ip route add 192.168.1.0/24 dev eth1 table 10
[root@localhost ~]# ip route add default via 192.168.1.254 table 10
[root@localhost ~]#
[root@localhost ~]# ip route show table 10
192.168.1.0/24 dev eth1 scope link
default via 192.168.1.254 dev eth1
[root@localhost ~]#
```

（3）删除路由：可以使用 ip 命令来方便地删除路由，使用以下示例来说明如何删除路由。首先将路由表 10 的内容显示出来，可以看到路由表 10 中当前有两条路由，接着使用 ip route del 命令删除默认路由，在此别忘了指定所要删除的是路由表 10，否则默认会删除路由表 main 的默认路由，接着使用 ip route show 命令查看路由表 10，此时路由表 10 的默认路由已经不存在了，再次使用 ip route del 命令删除 192.168.122.0/24 的路由，最后可以看到路由表 10 中已经没有任何路由了。完整的命令如下：

```
[root@localhost ~]# ip route show table 10
192.168.1.0/24 dev virbr0 scope link
default via 192.168.1.254 dev eth1
[root@localhost ~]#
[root@localhost ~]# ip route del default table 10
[root@localhost ~]#
[root@localhost ~]# ip route show table 10
192.168.1.0/24 dev virbr0 scope link
[root@localhost ~]#
[root@localhost ~]# ip route del 192.168.1.0/24 table 10
```

```
[root@localhost ~]#
[root@localhost ~]# ip route show table 10
[root@localhost ~]#
```

ip rule 路由策略数据库管理命令，/etc/iproute2/rt_tables 中的 table id 和 table name 是对应关系，如果不使用 table name 而只使用 table id，则 rt-tables 文件应该可以不修改。

ip rule add 添加规则可以使用 priority、order 或 preference（或者三者的简写分别为 pri、ord、pref）来定义优先级，不然第 1 条 rule 的优先级为 32765。

需要注意的是 ip rule 规则是以优先级为唯一的 key，也就说只要求优先级不能一样，具体的规则内容却可以一样，这就为使用 ip rule del 命令进行删除提供了便利（删除时指定优先级即可）。

ip rule add 的 from 及 to 都是以 PREFIX 为参数的，而 PREFIX 可以是 IP 地址也可以是 IP 地址段。

3.6 高级自动化运维工具 Ansible

3.6.1 Ansible 简介

Ansible 是新出现的自动化运维工具，基于 Python 研发。结合了众多老牌运维工具的优点，实现了批量操作系统配置、批量程序部署、批量运行命令等功能。仅需要在管理工作站上安装 Ansible 程序配置被管控主机的 IP 信息，被管控的主机无客户端。Ansible 应用程序存在于 epel（第三方社区）源，依赖于很多 Python 组件，主要包括以下组件。

（1）连接插件 connection plugins：负责和被监控端实现通信。

（2）host inventory：指定操作的主机，是一个配置文件里面定义监控的主机。

（3）各种模块的核心模块、command 模块、自定义模块。

（4）借助于插件完成记录日志邮件等功能。

（5）playbook：当剧本执行多个任务时，可以让节点一次性运行多个任务。

以 CentOS 7. x 系统安装 Ansible 为例，yum 安装是很常用的安装方式，需要先安装一个 epel-release 包，然后安装 Ansible，命令如下：

```
yum install epel-release -y
yum install ansible -y
```

安装目录如下（yum 安装）。

配置文件目录：/etc/ansible/。

执行文件目录：/usr/bin/。

Lib 库依赖目录：/usr/lib/pythonX. X/site-packages/ansible/。

Help 文档目录：/usr/share/doc/ansible-X. X. X/。

Man 文档目录：/usr/share/man/man1/。

3.6.2　Ansible 特性

(1) 模块化设计，调用特定的模块来完成特定任务，本身是核心组件，短小精悍。

(2) 基于 Python 语言实现，由 Paramiko(Python 的一个可并发连接 SSH 主机功能库)，PyYAML 和 Jinja2(模板化)共 3 个关键模块实现。

(3) 部署简单，无客户端工具。

(4) 主从模式工作。

(5) 支持自定义模块功能。

(6) 支持 playbook，连续任务按先后设置顺序完成。

(7) 期望每个命令具有幂等性。

Ansible 与其他配置管理软件的对比，以及技术特性比较见表 3-5 所示。

表 3-5　配置管理软件技术特性对比

项目	Puppet	Saltstack	Ansible
开发语言	Ruby	Python	Python
是否有客户端	是	是	否
是否支持二次开发	不支持	支持	支持
服务器与远程机器是否相互验证	是	是	是
服务器与远程机器通信是否加密	是，标准 SSL 协议	是，使用 AES 加密	是，使用 OpenSSH
是否提供 Web UI	提供	提供	提供，商业版本
配置文件格式	Ruby 语法	YAML	YAML
命令行执行	不支持，但可以通过配置模块实现	支持	支持

3.6.3　Ansible 架构

Ansible 的架构如图 3-1 所示，每个模块的核心作用如下。

(1) Ansible Core：ansible 自身的核心模块。

(2) Host Inventory：主机库，定义可管控的主机列表。

(3) Connection Plugins：连接插件，一般默认基于 SSH 协议连接。

(4) Modules：Core Modules(自带模块)、Custom Modules(自定义模块)。

(5) Playbooks：剧本，按照所设定编排的顺序执行完成安排任务。

3.6.4　配置文件

Ansible 配置文件查找顺序，ansible 与其他的服务在这一点上有很大不同，这里的配置文件查找是指从多个地方查找，顺序如下：

(1) 检查环境变量 ANSIBLE_CONFIG 指向的路径文件，例如 export ANSIBLE_CONFIG=/etc/ansible.cfg。

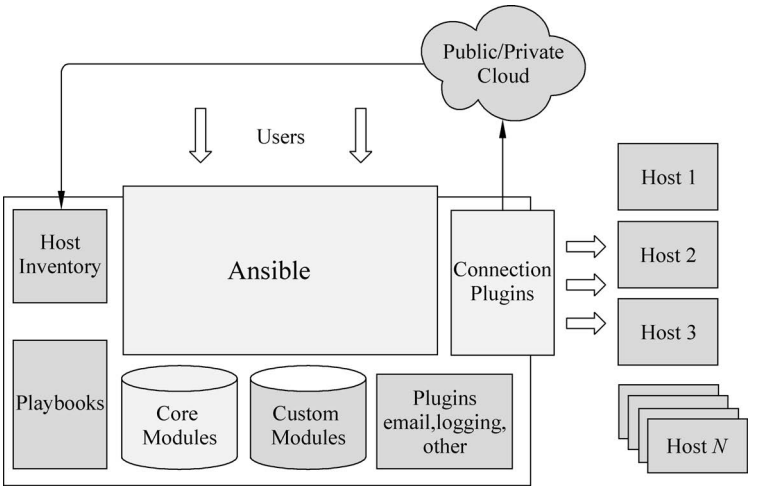

图 3-1 Ansible 的架构

(2) ~/.ansible.cfg 检查当前目录下的 ansible.cfg 配置文件。

(3) /etc/ansible.cfg 检查 etc 目录的配置文件。

Ansible 的配置文件为/etc/ansible/ansible.cfg,ansible 有许多参数,下面列出一些常见的参数:

```
inventory = /etc/ansible/hosts    #这个参数表示资源清单 inventory 文件的位置
library = /usr/share/ansible      #指向存放 Ansible 模块的目录,支持多个目录方式,只要用
                                  #冒号(:)隔开就可以
forks = 5                         #并发连接数,默认为 5
sudo_user = root                  #设置默认执行命令的用户
remote_port = 22    #指定连接被管节点的管理端口,默认为 22 端口,建议修改,能够更加安全
host_key_checking = False         #设置是否检查 SSH 主机的密钥,值为 True/False
                                  #关闭后第 1 次连接不会提示配置实例
timeout = 60                      #设置 SSH 连接的超时时间,单位为秒
log_path = /var/log/ansible.log   #指定一个存储 Ansible 日志的文件(默认不记录日志)
```

(1) ansible 应用程序的主配置文件:/etc/ansible/ansible.cfg。

(2) Host Inventory 定义管控主机:/etc/ansible/hosts。

遵循 INI 风格;中括号中的字符是组名;一个主机可同时属于多个组。

【示例 3-1】 Ungrouped hosts,specify before any groupheaders。直接在任何组的头部指定,不属于任何组的主机,内容如下:

```
green.example.com
blue.example.com
192.168.100.1
192.168.100.10
```

【示例 3-2】 A collection of hosts belonging to the'webservers' group。一批主机属于

一个组，例如定义为 webservers 组，内容如下：

```
[webservers]
alpha.example.org
beta.example.org
192.168.1.100
192.168.1.110
```

默认为以 root 用户执行，但是基于 SSH 连接操作要多次输入密码，为了方便可以使用基于 SSH 密钥方式进行认证。

3.6.5　Ansible 应用程序命令

ansible-doc 命令用于获取模块列表，以及模块使用格式。

ansible-doc-l：获取列表。

ansible-doc-s module_name：获取指定模块的使用信息。

Ansible 命令格式：ansible < host-pattern > [-f forks] [-m module_name] [-a args] < host-pattern >。

参数含义见表 3-6 所示。

表 3-6　Ansible 程序命令参数含义

参　数	含　义
< host-pattern >	指明管控主机，以模式形式表示或者直接给定 IP，必须事先定义在文件中；all 表示所有
[-f forks]	指明每批管控多少主机，默认 5 个主机为一批次
[-m module_name]	使用何种模块管理操作，所有的操作都需要通过模块来指定
[-a args]	指明模块专用参数；args 一般为键-值对格式 注意：command 模块的参数不是键-值对格式，而是直接给出要执行的命令

command 模块的参数不是键-值对格式，而是直接给出要执行的命令即可。

< host-pattern >默认读取/etc/ansible/hosts，也可以指明自定义文件路径。

-iPATH，--inventory＝PATH：指明使用的 host inventory 文件路径。

常用模块(module_name)如下。

(1) command：默认模块，可省略。在远程主机上操作命令，执行形式如下：

```
-a 'COMMAND'
```

command 模块的参数不是键-值对格式，直接给出要执行的命令，代码如下：

```
[root@localhost ~]# ansible all -m command -a 'ifconfig'
```

(2) user：在远程主机上操作命令，执行形式如下：

```
-a 'name= state={present(创建)|absent(删除)} force=(是否强制操作删除家目录) system=
uid= shell= home='
```

命令如下：

```
[root@localhost ~]# ansible all -m user -a 'name=ansible state=present'
```

(3) group：在远程主机上操作命令，执行形式如下：

```
-a 'name= state={present|absent} gid= system=(系统组)'
```

命令如下：

```
[root@localhost ~]# ansible all -m group -a 'name=mygroup state=presentsystem=true'
```

(4) cron：在远程主机上操作命令，执行形式如下：

```
-a 'name= state= minute= hour= day= month= weekday= job='
```

命令如下：

```
[root@localhost ~]# ansible all -m cron -a 'name='Time' state=presentminute='*/5' job='/
usr/sbin/ntpdate 172.168.0.1 &> /dev/null''
```

(5) ping：无参数，命令如下：

```
[root@localhost ~]# ansible all -m ping
```

(6) file：文件管理，在远程主机上操作命令，执行形式如下：

```
-a 'path= mode= owner= group= state={file|directory|link|hard|touch|absent} src=
(link,连接至何处)'
```

命令如下：

```
[root@localhost ~]# ansible all -m file -a 'path=/tmp/testdirstate=directory'
[root@localhost ~]# ansible all -m file -a 'path=/tmp/test.txt \
state=touchmod=600 owner=user1'
```

(7) copy：在远程主机上操作命令，执行形式如下：

```
-a 'dest=(远程主机上路径) src=(本地主机路径) content=(直接指明内容) owner= group=
mode='
```

命令如下：

```
[root@localhosttmp]# ansible web -m copy -a \
'src=/etc/yum.repos.d/aliyun.repodest=/etc/yum.repos.d/'
```

(8) template：在远程主机上操作命令，执行形式如下：

```
-a 'dest= src=\'#\'" content= owner= group= mode='
```

(9) yum：在远程主机上操作命令，执行形式如下：

```
-a 'name= conf_file=(指明配置文件) state={present|latest|absent}
enablerepo= disablerepo='
```

命令如下：

```
[root@localhost ~]#ansible all -m yum 'name=httpd state=present'
```

(10) service：在远程主机上操作命令，执行形式如下：

```
-a 'name= state={started|stopped|restarted} enabled=(是否开机时自动启动) runlevel='
```

命令如下：

```
[root@localhost ~]#ansible all -m service -a 'name=httpd state=started'
```

(11) shell：在远程主机上操作命令，执行形式如下：

```
-a 'COMMAND'运行 Shell 命令
```

命令如下：

```
[root@localhost ~]#ansible all -m shell -a echo "123456789" |passwd --stdin user1'
```

(12) script：在远程主机上操作命令，执行形式如下：

```
-a '/PATH/TO/SCRIPT'运行脚本
```

命令如下：

```
[root@localhost ~]#ansible all -m script -a '/tmp/a.sh'
```

(13) setup：获取指定主机的 facts 变量，在远程主机上操作命令，没有参数，命令如下：

```
ansible 192.168.0.102 -m setup
```

3.6.6　Playbooks 剧本

1. Playbook 组织格式

Playbook 组织格式：YAML 语言格式。

Playbooks 是 ansible 更强大的配置管理组件，实现基于文本文件编排执行的多个任务，并且可多次重复执行。

YAML 官方介绍为 YAML Arn't Markup Language；Yet Another Markup Language。翻译过来就是，类似于半结构化数据，声明式配置；可读性较高的用来表达资料序列的格式，易于与脚本语言交互。

语法格式遵循以下规则：

(1) 任何数据结构都用缩进来标识，可以嵌套。

(2) 每行是一个键值数据 key：value，用冒号隔开。若想在一行标识，则需要用{ }和逗号分隔格式。

(3) 列表用“-”标识。

2. inventory 参数

主机库 ssh 参数设置，ansible 基于 SSH 连接 inventory 中指定的远程主机时，将以此处的参数指定的属性进行，如表 3-7 所示。

表 3-7 ansible 进行 SSH 连接时指定参数

参 数	含 义
ansible_ssh_port	指定 SSH 端口
ansible_ssh_user	指定 SSH 用户
ansible_ssh_pass	指定 SSH 用户登录时使用的是认证密码，明文密码不安全
ansible_sudo_pass	指明 sudo 时候的密码

实例内容如下：

```
[websrvs]
192.168.0.101 ansible_ssh_port = 22 ansible_ssh_user = root
ansible_ssh_pass = root ansible_sudo_pass = root
192.168.0.102 ansible_ssh_port = 22 ansible_ssh_user = root
ansible_ssh_pass = root ansible_sudo_pass = root
```

在/etc/ansible/hosts 中直接定义连接时的密码不安全，一般建议基于 SSH 的密钥认证方式实现。

3. Playbooks

(1) 核心元素包含 Tasks(任务)、Variables(变量)、Templates(模板)、Handlers(处理器)、Roles(角色)。

(2) 在 playbooks 中定义任务，包含如下参数。

-name：task description 注释描述信息。

module_name：module_args 声明模块，定义 Ansible 模块参数。

举一个简单的例子，内容如图 3-2 所示。

(3) ansible-playbook 执行命令，命令格式如下：

```
ansible - playbook < filename.yml > ... [options]
```

举例运行如图 3-3 所示。

```
[root@localhost tmp]# vim 1.yml
[root@localhost tmp]# cat 1.yml
- hosts: web
  remote_user: root
  tasks:
        - name: install httpd
          yum: name=httpd state=present
        - name: install php
          yum: name=php state=present
        - name: start httpd
          service: name=httpd state=started enabled=true
[root@localhost tmp]#
```

图 3-2　Playbooks 定义任务示例

```
[root@localhost tmp]# ansible-playbook 1.yml

PLAY [web] *********************************************************************

GATHERING FACTS ****************************************************************
ok: [192.168.0.101]
ok: [192.168.0.103]
ok: [192.168.0.104]

TASK: [install httpd] **********************************************************
```

图 3-3　ansible-playbook 执行命令

(4) Playbook 变量，变量命名：由字母、数字和下画线组成，仅能以字母开头。

变量种类如下。

① facts：由远程主机发回的主机特有的属性信息，这些信息被保存在 Ansible 变量中；无须声明，可直接调用；

② 自定义变量：通过命令行传递，例如 ansible-playbook test. yml --extra-vars "host=www user=test" 通过 roles 传递；

③ 主机变量：定义在 inventory 中的主机之后的变量；直接传递给单个主机的变量。

直接定义在主机之后，内容如下：

```
[root@localhost ~]# vim /etc/ansible/hosts
[web]
192.168.0.101 host = mail
192.168.0.102
192.168.0.103
```

组变量：定义在 inventory 中的组变量(例如在默认的文件/etc/ansible/hosts 上编辑)，形式如下：

```
[group_name:vars]
var1 = value
var2 = value
```

组名要事先存在，代码如下：

```
[websrvs]
192.168.0.101
192.168.0.102
[websrvs:vars]
host = mail
```

变量使用示例，代码如下：

```
[root@localhost~]# vim useradd.yml
- hosts: websrvs
remote_user: root
vars:
username: testuser
password: xuding
tasks:
- name: add user
user: name = {{ username }} state = present
- name: set password
shell: /bin/echo {{ password }} |/usr/bin/passwd -- stdin {{ username }}
```

其中双花括号{{ }}用于调用变量。

变量的重新赋值，调用方法的形式如下：

```
ansible - playbook /PATH/TO/SOME_YAML_FILE { - eVARS| -- extra - vars = VARS}
```

举例说明如下：

```
[root@localhost ~]# ansible - playbook useradd.yml -- extra - vars "username = Ubuntu"
```

(5) playbook tasks 包含两种情况：

第 1 种情况为条件测试。在某 task 后面添加 when 子句即可实现条件测试功能；when 语句支持 Jinja2 语法；当使用 Red Hat 系列系统调用 yum 安装时，配置内容如下：

```
tasks:
- name: install web server package
yum: name = httpd state = present
when: ansible_os_family == "Red Hat"
```

第 2 种情况为迭代。在 task 中调用内置的 item 变量；在某 task 后面使用 with_items 语句来定义元素列表，编写内容如下：

```
tasks:
- name: add four users
user: name = {{ item }} state = present
with_items:
- testuser1
- testuser2
```

```
- testuser3
- testuser4
```

迭代中，列表中的每个元素可以为字典格式，编写内容如下：

```
- name: add two users
user: name = {{ item.name }} state = present groups = {{ item.groups }}
with_items:
- { name: 'testuser5', groups: 'wheel' }
- { name: 'testuser6', groups: 'root' }
```

(6) playbook handlers：处理器、触发器，只有其关注的条件满足时，才会触发执行的任务。

配置文件发生改变，触发重启服务，内容如下：

```
- hosts: websrvs
remote_user: root
tasks:
- name: install httpd
yum:name = httpd state = present
- name: install config file
copy: src = /root/httpd.confdest = /etc/httpd/conf/httpd.conf
notify: restart httpd
- name: start httpd service
service: name = httpd state = started
handlers:
- name: restart httpd
service: name = httpd state = restarted
```

4. Ansible Ad-Hoc 命令的使用

Ansible 系统由控制主机对被管节点的操作方式可分为两类，即 Ad-Hoc 和 Playbook。

Ad-Hoc 模式使用单个模块，支持批量执行单条命令。

Playbook 模式是 Ansible 的主要管理方式，也是 Ansible 功能强大的关键所在，Playbook 通过多个 Task 集合完成一类功能，如 Web 服务的安装部署、数据库服务器的批量备份等，可以简单地把 Playbook 理解为通过组合多条 Ad-Hoc 操作的配置文件。

通常以命令行的形式使用 Ansible 模块，或者将 Ansible 命令嵌入脚本中执行。Ansible 自带了很多模块，可以直接使用。当不知道如何使用这些模块时，可以使用 ansible-doc 命令获取帮助，例如使用 ansible-doc-l 命令可以显示所有自带的模块和相关简介，使用“ansible-doc 模块名”命令可以显示该模块的参数及用法等内容。

(1) Ansible Ad-Hoc 命令参数，可以使用 ansible-h 命令来列出所有的命令参数，下面列举了常用的一些参数，部分参数如果不指定，则将采用 ansible.cfg 文件中的设置值，或者采用原始默认值，详细参数如表 3-8 所示。

表 3-8 Ansible Ad-Hoc 主要命令参数

参　数	说　明
-V	输出详细信息
-VVVV	输出 Debug 信息
-i	指定 inventory 文件,默认值为/etc/ansible/hosts
-f	指定 fork 的进程个数,默认值为 5
-list-hosts	列出主机列表,并不会执行其他操作
-private-key=xxx	指定 SSH 连接使用的文件
-m	指定 module,默认为 command
-a	指定 module 的参数
-O	将输出内容精简至一行
-k	提示输入密码
-K	提示输入 sudo 密码,与-sudo 一起使用
-T	设置连接超时时间
-B	设置异步运行,并设置超时时长
-P	设置异步轮询时间
-t	指定输出信息的目录路径,结果文件以远程主机命名

(2) Ansible 常用的执行命令可以采用 4 种方式,第 1 种方式是利用 Command 模块在远程主机上执行命令,但 Command 模块不支持管道命令,例如,查看某个主机的日期,命令如下:

```
ansible ip -m command -a date -o
```

值得注意的是,Ansible 默认的模块是 Command,所以上面的命令可以简化,命令如下:

```
ansible ip -a date -o
```

第 2 种方式是利用 Shell 模块,切换到某个 Shell 执行远程主机上的 Shell/Python 脚本,或者执行命令,Shell 支持管道命令,功能较 Command 更强大更灵活,举例内容如下:

```
ansible ip -m shell -a 'bash /root/test.sh' -o
ansible ip -m shell -a 'echo "123456"|passwd --stdin root'
```

第 3 种方式是利用 Raw 模块,Raw 支持管道命令。Raw 有很多地方和 Shell 类似,但是如果使用老版本 Python(低于 2.4),则无法通过 Ansible 的其他模块执行命令,此种情况需要先用 Raw 模块远程安装 Python-sim-plejson 后才能受管控;又或者受管端是路由设备,因为没有安装 Python 环境,那就更需要使用 Raw 模块去管控了,命令如下:

```
ansible ip -m raw -a "cd /tmp;pwd"
```

第 4 种方式是利用 Script 模块,将 Ansible 中控端上的 Shell/Python 脚本传输到远端

主机上执行，即使远端主机没有安装 Python 也可以执行，有点类似 Raw 模块，但 Script 只能执行脚本，不能调用其他指令，并且不支持管道命令，命令如下：

```
ansible ip -m script -a '/root/test.sh' -o
```

removes 参数用来判断远端主机上是不是存在 test.sh 文件，如果存在，就执行管控机上的 test.sh 文件，如果不存在就不执行，命令如下：

```
ansible ip -m script -a 'removes=/root/test.sh /root/test.sh' -o
```

creates 用来判断远端主机上是不是存在 test.sh 文件，如果存在，就不执行，如果不存在，就执行管控机上的 test.sh 文件，命令如下：

```
ansible ip -m script -a 'creates=/root/test.sh /root/test.sh' -o
```

(3) Ansible 复制文件，常用的文件操作模块就是 copy 模块，它主要用于将本地或远程机器上的文件复制到远程主机上，其主要参数如表 3-9 所示。

表 3-9　copy 模块的主要命令参数

参　数	说　明
backup	指定远程主机上相同文件是否备份，yes 为备份，no 为不备份
dest	指定文件将被复制到远程主机的哪个目录中，dest 为必须参数
group	指定文件复制到远程主机后的属组
mode	指定文件复制到远程主机后的权限
owner	指定文件复制到远程主机后的属主
src	用于指定需要复制的文件或目录

将本地文件复制到远程主机，命令如下：

```
ansible ip -m copy -a 'src=/root/test.sh dest=/tmp/test.sh'
```

复制并修改文件的权限，命令如下：

```
ansible ip -m copy -a 'src=/root/test.sh dest=/tmp/test.sh mode=755'
```

复制并修改文件的属主，命令如下：

```
ansible ip -m copy -a 'src=/root/test.sh dest=/tmp/test.sh mode=755 owner=root'
```

复制文件前备份，命令如下：

```
ansible ip -m copy -a 'src=test.sh backup=yes dest=/tmp'
```

(4) Ansible 服务管理，在 Ansible Ad-Hoc 中，Service 模块可以帮助管理远程主机上的服务。例如，启动或停止远程主机中的某个服务，但是该服务本身必须能够通过操作系统的

管理服务的组件所管理，例如 RHEL 6 中默认通过 SysV 进行服务管理，RHEL 7 中默认通过 systemd 管理服务，如果该服务本身都不能被操作系统的服务管理组件所管理，则不能被 service 模块管理。该模块的几个主要命令参数如表 3-10 所示。

表 3-10 service 模块的主要命令参数

参 数	说 明
name	用于指定需要操作的服务名称
state	用于指定服务的状态启动：started；停止：stopped；重启：restarted；重加载：reloaded
enabled	用于指定是否将服务设置为开机启动项

启动服务，命令如下：

```
ansible all -m service -a "name=sshd state=started"
```

停止服务，命令如下：

```
ansible all -m service -a "name=sshd state=stopped"
```

开启服务自启动，命令如下：

```
ansible all -m service -a "name=sshd enable=yes"
```

(5) Ansible 安装包管理，在 Ansible Ad-Hoc 中，可以通过 yum 模块实现在远程主机上通过 yum 源管理软件包，包括安装、升级、降级、删除和列出软件包等。该模块的几个主要命令参数如表 3-11 所示。

表 3-11 yum 模块的主要命令参数

参 数	说 明
name	用于指定需要管理的软件包
state	用于指定软件包的状态
	安装：present/installed；安装最新版：latest；删除：absent/removed
enable repo	用于指定安装软件包时临时启用的 Yum 源
disable repo	用于指定安装软件包时临时禁用的 Yum 源

安装软件包，命令如下：

```
ansible all -m yum -a 'name=nginx state=installed'
```

卸载软件包，命令如下：

```
ansible all -m yum -a 'name=nginx state=removed'
```

临时启用 local yum 源安装最新版软件包，命令如下：

```
ansible all -m yum -a 'name=nginx state=latest enablerepo=local'
```

(6) Ansible 用户管理，在 Ansible Ad-Hoc 中，可以通过 user 模块帮助管理远程主机上的用户，例如创建用户、修改用户、删除用户、为用户创建密钥对等操作。该模块的几个常用参数如表 3-12 所示。

表 3-12　user 模块的主要命令参数

参　数	说　明
name	用于指定需要管理的软件包
state	用于指定软件包的状态
	安装：present/installed；安装最新版：latest；删除：absent/removed
enable repo	用于指定安装软件包时临时启用的 yum 源
disable repo	用于指定安装软件包时临时禁用的 yum 源

增加用户、组和密码，命令如下：

```
ansible ip -m group -a "name=testg"
ansible ip -m user -a "name=test group=testg password=123456
home=/home/test"
```

删除用户和用户主目录，命令如下：

```
ansible ip -m user -a "name=test. state=absent remove=yes"
```

5. Ansible Facts 的使用

Facts 组件是 Ansible 用于采集被管主机设备信息的一个功能，当 Ansible 采集 Fact 时，它会收集被管主机的各种详细信息：CPU 架构、操作系统、IP 地址、内存信息、磁盘信息等，这些信息保存在被称作 Fact 的变量中。Ansible 使用一个名为 Setup 的特殊模块实现对 Fact 进行收集，在 Playbook 中默认会调用这个模块进行 Fact 收集，在命令行中可以通过 ansible ip-m setup 命令进行手动收集，整个 Facts 信息被包装在 JSON 格式的数据结构中，Ansible Facts 是最上层的值，如图 3-4 所示。

Facts 还支持通过 filter 参数来查看指定信息，例如只查看远端主机的操作系统和版本，如图 3-5 所示。

查看远端主机的 CPU 和内存大小，如图 3-6 所示。

查看远端主机的各文件系统大小和剩余容量，如图 3-7 所示。

在 Playbook 中，Facts 组件默认会收集很多主机的基础信息，可以通过前面的 Facts 缓存机制，将这些信息缓存到本地目录或者内存数据库中，在做配置管理时进行引用，也可以用来将获取的主机基础信息自动同步到 CMDB 中，实现基础信息的自动采集功能。下面通过演示，说明如何通过 ansible-cmdb 插件，实现将远端主机 CPU 自动同步至外部 CMDB 系统中。

首先，需要安装 ansible-cmdb 插件，下载链接如下：

```
https://files.pythonhosted.org/packages/37/1b/1fcff0a38a4e07d9d3f75113494
ec0b25fd271b650bda52907ae1a80cbfb/ansible-cmdb-1.27.tar.gz
```

```
ansible X.X.X.2 -m setup
X.X.X.2 | SUCCESS => {
    "ansible_facts": {
        "ansible_all_ipv4_addresses": [
            "X.X.X.2"
        ],
        "ansible_all_ipv6_addresses": [
            "fe80::f816:3eff:fe1e:67f3"
        ],
        "ansible_apparmor": {
            "status": "disabled"
        },
        "ansible_architecture": "x86_64",
        "ansible_bios_date": "01/01/2011",
        "ansible_bios_version": "0.5.1",
        "ansible_cmdline": {
            "BOOT_IMAGE": "/vmlinuz-3.10.0-327.el7.x86_64",
            "biosdevname": "0",
            "console": "ttyS0,115200n8",
            "crashkernel": "auto",
            "net.ifnames": "0",
            "rd.lvm.lv": "rhel/root",
            "ro": true,
            "root": "/dev/mapper/rhel-root"
        },
```

图 3-4　ansible facts 采集主机信息

```
ansible X.X.X.2 -m setup -a "filter=ansible_system" -o
X.X.X.2 | SUCCESS => {"ansible_facts": {"ansible_system": "Linux"}, "changed": false}
ansible X.X.X.2 -m setup -a "filter=ansible_distribution" -o
X.X.X.2 | SUCCESS => {"ansible_facts": {"ansible_distribution": "RedHat"}, "changed": false}
ansible X.X.X.2 -m setup -a "filter=ansible_distribution_version" -o
X.X.X.2 | SUCCESS => {"ansible_facts": {"ansible_distribution_version": "7.2"}, "changed": false}
```

图 3-5　ansible facts 过滤主机操作系统信息

```
ansible X.X.X.2 -m setup -a "filter=ansible_processor_count" -o
X.X.X.2 | SUCCESS => {"ansible_facts": {"ansible_processor_count": 4}, "changed": false}
ansible X.X.X.2 -m setup -a "filter=ansible_memtotal_mb" -o
X.X.X.2 | SUCCESS => {"ansible_facts": {"ansible_memtotal_mb": 15887}, "changed": false}
```

图 3-6　ansible facts 过滤主机 CPU 和内存信息

其次，开始安装 ansible-cmdb 插件：

```
gzip -dc ansible-cmdb-1.27.tar.gz|tar -xvf -
cd ansible-cmdb-1.27
python setup.py install
```

生成所有主机的 Facts 信息并用 filter 过滤出主机 CPU 值：

```
ansible all -m setup -t /tmp/factout
ansible all -m setup -a "filter=ansible_processor_count" -t /tmp/cpu
```

通过 ansible-cmdb 插件以 CSV 或 SQL 格式输出 IP 地址和 CPU 值：

```
ansible X.X.X.2 -m setup  -a "filter=ansible_mounts"
X.X.X.2 | SUCCESS => {
    "ansible_facts": {
        "ansible_mounts": [
            {
                "device": "/dev/mapper/rhel-root",
                "fstype": "xfs",
                "mount": "/",
                "options": "rw,relatime,attr2,inode64,noquota",
                "size_available": 94723239936,
                "size_total": 106819743744,
                "uuid": "517bc764-fd26-4027-a301-6c5e29557f7c"
            },
            {
                "device": "/dev/vda1",
                "fstype": "xfs",
                "mount": "/boot",
                "options": "rw,relatime,attr2,inode64,noquota",
                "size_available": 354861056,
                "size_total": 520794112,
                "uuid": "2c98f0c5-9dbb-4cac-871c-8e15e802a173"
            },
        ]
    },
    "changed": false
}
```

图 3-7　ansible facts 过滤主机文件系统信息

```
PATH = /usr/local/bin: $ PATH
ansible - cmdb  - t csv  - c name,cpus /tmp/cpu >/tmp/cpu.csv
ansible - cmdb  - t sql /tmp/cpu >/tmp/cpu.sql
```

以 CSV 格式或者 SQL 格式将信息导入外部 CMDB 系统中(根据支持方式灵活选用)，这里以 postgres 数据库为例，通过将 CSV 格式转换为 SQL 格式导入外部 CMDB 系统，示例如图 3-8 所示。

```
cpuinfo=$(cat /tmp/cpu.csv)
for cpu_info in $cpuinfo
do
ip=$(echo "$cpu_info" |awk -F"," '{print $1}'|sed 's/"//g')
cpu=$(echo "$cpu_info" |awk -F"," '{print $2}'|sed 's/"//g')
echo "update \"VirtualMachine\" set \"VCPU\"='$cpu' where \"IP\"='$cpuip' and \"Status\"='A';"
>>/tmp/cpu_info.sql
done
su - postgres -c "export PGPASSWORD=postgres;/opt/PostgreSQL/9.3/bin/psql -d cmdbuild -f
/tmp/cpu_info.sql"
```

图 3-8　将数据导入外部 CMDB 系统

当然，以上仅仅是简单的示例，可以利用 Ansible Fact 的功能，实现更加复杂的 CMDB 自动化采集功能。

6. Ansible 常用参数列表

(1) inventory 内置参数如表 3-13 所示。

表 3-13 inventory 内置参数

参　考	解　释	例　子
ansible_ssh_host	将要连接的远程主机名与想要设定的主机的别名不同,可通过此变量设置	ansible _ ssh _ host = 192.169.1.123
ansible_ssh_port	SSH 端口号。如果不是默认的端口号,则通过此变量设置	ansible _ ssh _ port = 5000
ansible_ssh_user	默认的 SSH 用户名	ansible _ ssh _ user = cxpadmin
ansible_ssh_pass	SSH 密码(这种方式并不安全,强烈建议使用 --ask-pass 或 SSH 密钥)	ansible _ ssh _ pass = '123456'
ansible_sudo_pass	sudo 密码(这种方式并不安全,强烈建议使用 --ask-sudo-pass)	ansible_ sudo _ pass = '123456'
ansible_sudo_exe	sudo 命令路径(适用于 1.8 及以上版本)	ansible _ sudo _ exe = /usr/bin/sudo
ansible_connection	与主机的连接类型。例如 local、ssh 或者 paramiko。Ansible 1.2 以前默认使用 paramiko. 1.2,以后默认使用 'smart','smart' 方式会根据是否支持 ControlPersist 来判断'ssh' 方式是否可行	ansible _ connection = local
ansible_ssh_private_key_file	SSH 使用的私钥文件。适用于有多个密钥,而不想使用 SSH 代理的情况	ansible_ ssh _ private _ key_file=/root/key
ansible_shell_type	目标系统的 Shell 类型。在默认情况下,命令的执行使用 'sh' 语法,可设置为 'csh' 或 'fish'	ansible_ shell _ type = zsh
ansible_python_interpreter	目标主机的 Python 路径。适用于的情况:系统中有多个 Python,或者命令路径不是"/usr/bin/python",例如\ * BSD,或者 /usr/bin/python 不是 2.X 版本的 Python。不使用 "/usr/bin/env" 机制,因为这要求远程用户的路径设置正确,并且要求 "python" 可执行程序名不可为 python 以外的名字(实际有可能名为 python2.6)	ansible_python_ interpreter=/usr/ bin/python2.6
ansible_ * _interpreter	定义其他语言解释器	ansible_ * _interpreter= /usr/bin/ruby
ansible_sudo	定义 sudo 用户	ansible_sudo= cxpadmin

从 Ansible 2.0 开始,ansible_ssh_user、ansible_ssh_host、ansible_ssh_port 已经改变为 ansible_user、ansible_host、ansible_port。具体可参考官网信息,网址如下:

```
http://docs.ansible.com/ansible/latest/intro_inventory.html
```

(2) copy 模块:在 Ansible 里的角色就是把 Ansible 执行机器上的文件复制到远程节

点上，是与 fetch 模块相反的操作。常用模块参数如表 3-14 所示。

表 3-14　copy 模块的主要命令参数及说明

参　数　名	说　　明
src	用于定位 ansible 执行的机器上的文件，需要绝对路径。如果复制的是文件夹，则文件夹会被整体复制，如果结尾是“/”，则只有文件夹内的内容会被复制过去。一切感觉很像 rsync
content	用来替代 src，用于将指定文件的内容复制到远程文件内
dest	用于定位远程节点上的文件，需要绝对路径。如果 src 指向的是文件夹，则这个参数也必须指向文件夹
backup	备份远程节点上的原始文件，在复制之前。如果发生意外，则原始文件还能使用
directory_mode	这个参数只能用于复制文件夹，这个参数设定后，文件夹内新建的文件会被复制，而老旧的文件不会被复制
follow	当复制的文件夹内有 link 时，复制过去的文件也会有 link
force	默认为 yes，会覆盖远程的内容不一样的文件(可能文件名一样)。如果是 no，就不会复制文件
group	设定一个群组拥有复制到远程节点的文件权限
mode	等同于 chmod，参数可以为“u+rwx or u=rw,g=r,o=r”
owner	设定一个用户拥有复制到远程节点的文件权限

将文件复制到测试主机，示例命令如下：

```
[root@node1 ansible]# echo "test " >> /root/install.log
[root@node1 ansible]# ansible testservers -m copy -a \
'src=/root/install.log dest=/tmp/install.log owner=testuser \
group=testgroup backup=yes'

192.168.100.132 | success >> {
"backup_file": "/tmp/install.log.2016-02-25@16:01:26~",
"changed": true,
"checksum": "b5da7af32ad02eb98f77395b28f281a965b4c1f5",
"dest": "/tmp/install.log",
"gid": 1100,
"group": "testgroup",
"md5sum": "d39956add30a18019cb5ad2381a0cd43",
"mode": "0644",
"owner": "testuser",
"size": 9464,
"src": "/root/.ansible/tmp/ansible-tmp-1456387285.87-
128685659798967/source",
"state": "file",
"uid": 1000
}

192.168.100.131 | success >> {
"backup_file": "/tmp/install.log.2016-02-25@16:01:26~",
"changed": true,
"checksum": "b5da7af32ad02eb98f77395b28f281a965b4c1f5",
```

```
    "dest": "/tmp/install.log",
    "gid": 1100,
    "group": "testgroup",
    "md5sum": "d39956add30a18019cb5ad2381a0cd43",
    "mode": "0644",
    "owner": "testuser",
    "size": 9464,
    "src": "/root/.ansible/tmp/ansible-tmp-1456387285.86-
    134452201968647/source",
    "state": "file",
    "uid": 1000
    }
```

如果使用复制命令将文件夹复制到目标机器，则会发现有文件的目录复制成功，空的目录没有复制过去，因为没有文件的空目录会被忽略。copy 模块常用返回值参数及含义如表 3-15 所示。

表 3-15 copy 模块的返回值参数及含义

参数名	参数说明	返回值	返回值类型	样例
src	位于 Ansible 执行机上的位置	changed	string	/home/httpd/.ansible/tmp/ansible-tmp-1423796390.97-147729857856000/source
backup_file	将原文件备份	changed and if backup=yes	string	/path/to/file.txt.2015-02-12@22:09
uid	在执行后，拥有者的 ID	success	int	100
dest	远程节点的目标目录或文件	success	string	/path/to/file.txt
checksum	复制文件后的 checksum 值	success	string	6e642bb8dd5c2e027bf21dd923337cbb4214f827
md5sum	复制文件后的 md5 checksum 值	when supported	string	2a5aeecc61dc98c4d780b14b330e3282
state	执行后的状态	success	string	file
gid	执行后拥有文件夹、文件的群组 ID	success	int	100
mode	执行后文件的权限	success	string	0644
owner	执行后文件所有者的名字	success	string	httpd
group	执行后文件所有群组的名字	success	string	httpd
size	执行后文件大小	success	int	1220

(3) shell 模块,它负责在被 Ansible 控制的节点(服务器)执行命令。shell 模块是通过/bin/sh 执行的,所以 shell 模块可以执行任何命令,就像在本机执行一样。常用参数如表 3-16 所示。

表 3-16　shell 模块的主要命令参数及说明

参　数	说　明
chdir	跟 command 一样,运行 shell 之前通过 cd 命令转到某个目录
creates	跟 command 一样,如果某个文件存在,则不运行 shell 模块
removes	跟 command 一样,如果某个文件不存在,则不运行 shell 模块

【示例 3-3】 让所有节点运行 somescript.sh 文件并把 log 输出到 somelog.txt 文件,命令如下:

```
$ ansible -i hosts all -m shell -a "sh somescript.sh >> somelog.txt"
```

【示例 3-4】 先进入 somedir/目录,再在 somedir/目录下让所有节点运行 somescript.sh 文件并把 log 输出到 somelog.txt 文件,命令如下:

```
$ ansible -i hosts all -m shell -a "somescript.sh >> somelog.txt"
chdir=somedir/
```

【示例 3-5】 先通过 cd 命令转到某个需要编译的目录,执行 configure,然后编译及安装,命令如下:

```
$ ansible -i hosts all -m shell -a "./configure && make && make install" chdir=/xxx/yyy/
```

(4) command 模块用于运行系统命令。不支持管道符和变量符(<、>、|、& 等),如果要使用这些,则可以使用 shell 模块。在使用 Ansible 时,默认的模块是-m command,从而使模块的参数不需要填写,直接使用即可。常用的参数如表 3-17 所示。

表 3-17　command 模块的主要命令参数及说明

参　数	说　明
chdir	运行 command 命令前先通过 cd 命令转到这个目录
creates	如果这个参数对应的文件存在,就不运行 command
executable	将 shell 切换为 command 执行,这里的所有命令需要使用绝对路径
removes	如果这个参数对应的文件不存在,就不运行 command

【示例 3-6】 ansible 命令调用 command 进行关机,命令如下:

```
ansible -i hosts all -m command -a "/sbin/shutdown -t now"
```

ansible 命令行调用-m command-a 表示使用参数,引号内的内容为执行的 command 命令,该命令为关机,对应的节点(192.168.10.12 和 127.152.112.13)都会执行关机操作。

【示例 3-7】 利用 creates 参数,判断/path/to/database 文件是否存在,如果存在就跳过 command 命令,如果不存在就执行 command 命令,示例命令如下:

```
ansible -i hosts all -m command -a "/usr/bin/make_database.sh arg1 arg2 creates=/path/to/
database"
```

(5) raw 模块的功能与 shell 和 command 类似,但 raw 模块运行时不需要在远程主机上配置 Python 环境。

【示例 3-8】 在 10.1.1.113 节点上运行 hostname 命令,命令如下:

```
ansible 10.1.1.113 -m raw -a 'hostname|tee'
```

(6) fetch 模块,文件拉取模块主要用于将远程主机中的文件复制到本机中,和 copy 模块的作用刚刚相反,并且在保存时使用 hostname 进行保存,当文件不存在时,会出现错误,除非将选项 fail_on_missing 设置为 yes,常用的参数见表 3-18 所示。

表 3-18 fetch 模块的主要命令参数及说明

参 数	说 明
dest	用来存放文件的目录,例如存放目录为 backup,源文件名称为/etc/profile,在主机 pythonserver 中,那么存放文件的目录为/backup/pythonserver/etc/profile
fail_on_missing	当源文件不存在时,标识为失败
flat	允许覆盖默认行为从 hostname/path 到/file 的,如果 dest 以"/"结尾,则它将使用源文件的基础名称
src	在远程拉取的文件,并且必须是一个文件,不能是目录
validate_checksum	当文件 fetch 之后进行 MD5 检查

【示例 3-9】 fetch 一个文件并保存,src 表示为远程主机上需要传送的文件路径,dest 表示本机上的路径,传送过来的文件是按照 IP 地址进行分类的,并且路径是源文件的路径。在拉取文件时,拉取的必须是文件,不能拉取文件夹,示例命令如下:

```
[root@ansibleserver ~]# ansible pythonserver -m fetch -a "src=/root/123 dest=/root"
SSH password:
192.168.1.60 | success >> {
    "changed": true,
    "dest": "/root/192.168.1.60/root/123",
    "md5sum": "31be5a34915d52fe0a433d9278e99cac",
    "remote_md5sum": "31be5a34915d52fe0a433d9278e99cac"
}
```

【示例 3-10】 指定路径目录进行保存。在使用参数 flat 时,如果 dest 的后缀名为"/",就会保存在目录中,然后直接保存为文件名;当 dest 后缀不为"/"时,那么就会直接保存为目录 kel 的文件。主要在于 dest 是否以"/"结尾,从而来区分这是个目录还是路径,示例命令如下:

```
[root@ansibleserver ~]# ansible pythonserver -m fetch -a "src=/root/Ssh.py dest=/root/
kel/ flat=yes"
SSH password:
```

```
192.168.1.60 | success >> {
    "changed": true,
    "dest": "/root/kel/Ssh.py",
    "md5sum": "63f8a200d1d52d41f6258b41d7f8432c",
    "remote_md5sum": "63f8a200d1d52d41f6258b41d7f8432c"
}
```

(7) file 模块,主要用来设置文件、链接、目录的属性,或者移除文件、链接、目录,很多其他模块也会包含这种作用,例如 copy、assemble 和 template。常用的参数如表 3-19 所示。

表 3-19　file 模块的主要命令参数及说明

参　数	说　明
follow	这个标识说明这是系统链接文件,如果存在,则应该遵循
force	在两种情况下强制创建链接:源文件不存在(过会会存在);目标存在但是是文件(创建链接文件替代)
group	文件所属用户组
mode	文件所属权限
owner	文件所属用户
path	要控制文件的路径
recurse	当文件为目录时,是否进行递归设置权限
src	文件链接路径,只有状态为 link 时,才会设置,可以是绝对及相对不存在的路径
state	如果目录不存在,则会创建目录;如果文件不存在,则不会创建文件;如果是 link,则软链接会被创建或者修改;如果是 absent,则目录下的所有文件都会被删除,如果是 touch,则会创建不存在的目录和文件

【示例 3-11】 设置文件属性。文件路径为 path,表示文件路径,设定所属用户和所属用户组,权限为 0644。文件路径为 path,使用文件夹进行递归修改权限,使用的参数为 recurse,表示递归,示例命令如下:

```
[root@ansibleserver ~]# ansible pythonserver -m file -a "path=/root/123 owner=kel
group=kel mode=0644"
SSH password:
192.168.1.60 | success >> {
    "changed": true,
    "gid": 500,
    "group": "kel",
    "mode": "0644",
    "owner": "kel",
    "path": "/root/123",
    "size": 294,
    "state": "file",
    "uid": 500
}
[root@ansibleserver ~]# ansible pythonserver -m file -a "path=/tmp/kel/ owner=kel
group=kel mode=0644 recurse=yes"
SSH password:
```

```
192.168.1.60 | success >> {
    "changed": true,
    "gid": 500,
    "group": "kel",
    "mode": "0644",
    "owner": "kel",
    "path": "/tmp/kel/",
    "size": 4096,
    "state": "directory",
    "uid": 500
}
```

【示例 3-12】 创建目录。使用的参数主要是 state 为 directory，命令如下：

```
[root@ansibleserver ~]# ansible pythonserver -m file -a "path=/tmp/kel state=directory
mode=0755"
SSH password:
192.168.1.60 | success >> {
    "changed": true,
    "gid": 0,
    "group": "root",
    "mode": "0755",
    "owner": "root",
    "path": "/tmp/kel",
    "size": 4096,
    "state": "directory",
    "uid": 0
}
```

【示例 3-13】 修改权限。直接使用 mode 对权限进行修改，命令如下：

```
[root@ansibleserver ~]# ansible pythonserver -m file -a "path=/tmp/kel mode=0444"
SSH password:
192.168.1.60 | success >> {
    "changed": true,
    "gid": 0,
    "group": "root",
    "mode": "0444",
    "owner": "root",
    "path": "/tmp/kel",
    "size": 4096,
    "state": "directory",
    "uid": 0
}
```

【示例 3-14】 创建软连接。src 表示已经存在的文件，dest 表示创建的软连接的文件名，最后的 state 的状态为 link，示例命令如下：

```
root@ansibleserver tmp# ansible pythonserver -m file -a "src=/tmp/1
dest=/tmp/2 owner=kel state=link"
```

```
SSH password:
192.168.1.60 | success >> {
    "changed": true,
    "dest": "/tmp/2",
    "gid": 0,
    "group": "root",
    "mode": "0777",
    "owner": "kel",
    "size": 6,
    "src": "/tmp/1",
    "state": "link",
    "uid": 500
}
```

(8) yum 模块：Yum(全称为 Yellow dog Updater,Modified)是一个在 Fedora 和 Red Hat 及 CentOS 中的 Shell 前端软件包管理器,即安装包管理模块。常用的参数如表 3-20 所示。

表 3-20　yum 模块的主要命令参数及说明

参　数　名	说　　明
conf_file	设定远程 yum 执行时所依赖的 yum 配置文件
disable_gpg_check	在安装包前检查包,只会影响 state 参数为 present 或者 latest 时
list	只能由 ansible 调用,不支持 playbook
name	需要安装的包的名字,也能如此使用 name=python=2.7 安装 Python 2.7
state	用于描述安装包的最终状态,present/latest 用于安装包,absent 用于 remove 安装包
update_cache	用于在安装包前执行更新 list,只会影响 state 参数为 present/latest 时

安装 httpd 包,命令如下：

```
ansible host31 -m yum -a "name=httpd"
host31 | SUCCESS =>
"changed": true,
"msg": "",
"rc": 0,
"results": [ xxxxx ]
```

删除 httpd 包,命令如下：

```
ansible host31 -m yum -a "name=httpd state=absent"
host31 | SUCCESS =>
"changed": true,
"msg": "",
"rc": 0,
"results": [ xxxx ]
```

(9) service 模块,service 模块其实就是 Linux 下的 service 命令。用于 service 服务管

理，常用的参数如表 3-21 所示。

表 3-21 service 模块的主要命令参数及说明

参 数 名	说 明
enabled	启动 OS 后启动对应 service 的选项。使用 service 模块时，enabled 和 state 至少要有一个被定义
name	需要进行操作的 service 名字
state	service 最终操作后的状态

启动服务，命令如下：

```
ansible host31 -m service -a "name=httpd state=started"
host31 | SUCCESS => {
"changed": true,
"name": "httpd",
"state": "started"
}
```

停止服务，命令如下：

```
ansible host31 -m service -a "name=httpd state=stopped"
host31 | SUCCESS => {
"changed": true,
"name": "httpd",
"state": "stopped"
}
```

设置服务开机自启动，命令如下：

```
[root@host31 ~]# ansible host31 -m service -a "name=httpd enabled=yes state=
restarted"
host31 | SUCCESS => {
"changed": true,
"enabled": true,
"name": "httpd",
"state": "started"
}
```

(10) cron 模块，cron 模块用于管理计划任务，常用的参数如表 3-22 所示。

表 3-22 cron 模块的主要命令参数及说明

参 数 名	说 明
backup	对远程主机上的原任务计划内容修改之前做备份
cron_file	如果指定该选项，则用该文件替换远程主机上的 cron.d 目录下用户的任务计划
day	日(1～31，*，*/2，…)
hour	小时(0～23，*，*/2，…)
minute	分钟(0～59，*，*/2，…)

续表

参　数　名	说　　明
month	月(1～12，*，*/2，…)
weekday	周(0～7，*，…)
job	要执行的任务，依赖于 state=present
name	该任务的描述
special_time	指定什么时候执行，参数：reboot、yearly、annually、monthly、weekly、daily、hourly
state	确认该任务计划是创建还是删除
user	以哪个用户的身份执行

设置定时任务，示例命令如下：

```
ansible test -m cron -a 'name="a job for reboot" special_time=reboot job="/some/job.sh"'

ansible test -m cron -a 'name="yum autoupdate" weekday="2" minute=0 hour=12 user="root

ansible test -m cron -a 'backup="True" name="test" minute="0" hour="5,2" job="ls -alh >
/dev/null"'

ansible test -m cron -a 'cron_file=ansible_yum-autoupdate state=absent'
```

(11) user 模块请求的是 useradd、userdel、usermod 指令，常用的参数如表 3-23 所示。

表 3-23　user 模块的主要命令参数及说明

参　数　名	说　　明
home	指定用户的家目录，需要与 createhome 配合使用
groups	指定用户的属组
uid	指定用户的 uid
password	指定用户的密码
name	指定用户名
createhome	是否创建家目录 yes|no
system	是否为系统用户
remove	当 state=absent 时，remove=yes 表示连同家目录一起删除，等价于 userdel-r
state	是创建还是删除
shell	指定用户的 Shell 环境

当指定 password 参数时，不能使用明文密码，因为后面这一串密码会被直接传送到被管理主机的/etc/shadow 文件中，所以需要先对密码字符串进行加密处理，然后将得到的字符串放到 password 中。不同的发行版默认使用的加密方式可能会有区别，具体可以查看/etc/login.defs 文件确认，CentOS 6.5 版本使用的是 SHA512 加密算法。

在指定节点上创建一个用户名为 noLinux，组为 noLinux 的用户，示例命令如下：

```
ansible 10.1.1.113 -m user -a 'name=noLinux groups=noLinux state=present'
```

删除用户，示例命令如下：

```
ansible 10.1.1.113 -m user -a 'name=noLinux groups=noLinux state=absent remove=yes'
```

(12) group 模块：请求的是 groupadd、groupdel、groupmod 指令。

在所有节点上创建一个组名为 noLinux，gid 为 2014 的组，示例命令如下：

```
ansible all -m group -a 'gid=2014 name=noLinux'
```

(13) script 模块：将控制节点的脚本执行在被控节点上。

远程执行脚本，命令如下：

```
[root@host31 ~]#ansible host32 -m script -a /tmp/hello.sh
host32 | SUCCESS => {
"changed": true,
"rc": 0,
"stderr": "",
"stdout": "this is test from host32\r\n",
"stdout_lines": [ "this is test from host32" ->执行结果 ]
}
```

(14) get_url 模块：主要用于从 HTTP、FTP、HTTPS 服务器上下载文件，类似于 wget，常用的参数如表 3-24 所示。

表 3-24 get_url 模块的主要命令参数及说明

参 数 名	说 明
sha256sum	下载完成后进行 SHA256 检查
timeout	下载超时时间，默认为 10s
url	下载的 URL
url_password url_username	主要用于需要用户名和密码进行验证的情况
use_proxy	使用代理，代理需事先在环境变更中定义

将 http://10.1.1.116/favicon.ico 文件下载到指定节点的/tmp 目录下，示例命令如下：

```
ansible 10.1.1.113 -m get_url -a 'url=http://10.1.1.116/favicon.ico dest=/tmp'
```

(15) synchronize 模块，使用 rsync 同步文件，常用的参数如表 3-25 所示。

表 3-25 synchronize 模块的主要命令参数及说明

参 数 名	说 明
archive	归档，相当于同时开启 recursive(递归)、links、perms、times、owner、group、-D 选项，并且都为 yes，默认该项为开启
checksum	跳过检测 sum 值，默认关闭

续表

参　数　名	说　　明
compress	是否开启压缩
copy_links	复制链接文件，默认为 no，注意后面还有一个 links 参数
delete	删除不存在的文件，默认为 no
dest	目录路径
dest_port	dest_port：默认目录主机上的端口，默认为 22，使用的是 SSH 协议
dirs	传输目录不进行递归，默认为 no，即进行目录递归
rsync_opts	rsync 参数部分
set_remote_user	主要用于/etc/ansible/hosts 中定义或默认使用的用户与 rsync 使用的用户不同的情况
mode	push 或 pull 模块，push 模式一般用于从本机向远程主机上传文件，而 pull 模式用于从远程主机上取文件

将主控方/root/a 目录推送到指定节点的/tmp 目录下，示例命令如下：

```
ansible 10.1.1.113 -m synchronize -a 'src=/root/a dest=/tmp/compress=yes'
```

由于模块默认都是推送 push，因此，如果在使用拉取(pull)功能时，则可以使用 mode=pull 实现，将推送模式更改为拉取模式。

将 10.1.1.113 节点的/tmp/a 目录拉取到主控节点的/root 目录下，示例命令如下：

```
ansible 10.1.1.113 -m synchronize -a 'mode=pull src=/tmp/a dest=/root/'
```

由于模块默认启用了 archive 参数，该参数默认开启了 recursive、links、perms、times、owner、group 和-D 参数。如果将该参数设置为 no，则将停止很多参数，例如会导致目的递归失败，导致无法拉取。

(16) 其他模块用得不是特别多。

mount 模块：配置挂载点；

unarchive 模块：解压文件模块。

3.7　网络设备地址租约管理 DHCP

动态主机配置协议(Dynamic Host Configuration Protocol，DHCP)是基于客户机/服务器模式的一个简化主机 IP 地址分配管理的 TCP/IP 标准协议。DHCP 服务用于向计算机自动提供 IP 地址、子网掩码和路由信息。目的在于减轻 TCP/IP 网络的规划、管理和维护的负担，解决 IP 地址空间缺乏问题。

DHCP 是一个基于广播的协议，它由 BOOTP(Bootstrap Protocol，自举协议)发展而来，分为服务器端和客户端两部分。DHCP 租约提供了自动在 TCP/IP 网络上安全地分配和租用 IP 地址的机制，实现了统一规划和管理网络中的 IP 地址，提高了 IP 地址的使用率，

并方便了管理员的管理。DHCP 分配 IP 地址的过程如图 3-9 所示。

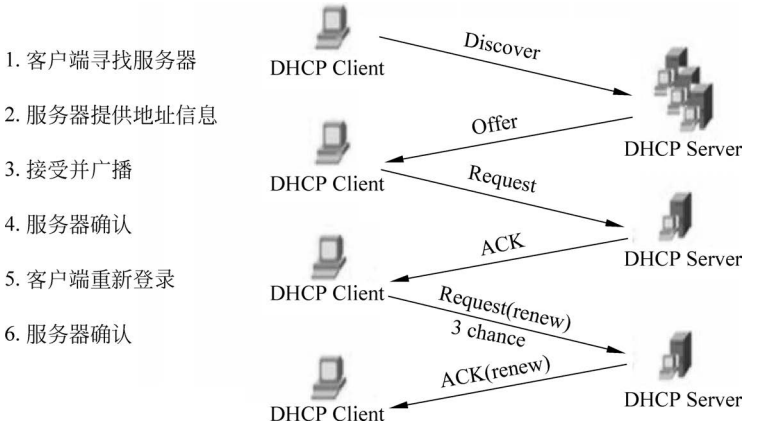

图 3-9 DHCP 分配 IP 地址的过程

3.7.1 私有网段

DHCP 服务一般配置的是私有网段，因此需要了解私有网段，现有 A、B、C 共 3 个地址段：

(1) A 类私有网段 10.0.0.0/8，可分配的 IP 地址范围为 10.0.0.0～10.255.255.255。

(2) B 类私有网段 172.16.0.0/12，可分配的 IP 地址范围为 172.16.0.0～172.31.255.255。

(3) C 类私有网段 192.168.0.0/16，可分配的 IP 地址范围为 192.168.0.0～192.168.255.255。

3.7.2 DHCP 报文种类

DHCP 一共有 8 种报文，各种类型报文种类及说明如表 3-26 所示。

表 3-26 DHCP 报文种类及说明

报文种类	说明
Discover(0x01)	DHCP 客户端在请求 IP 地址时并不知道 DHCP 服务器的位置，因此 DHCP 客户端会在本地网络内以广播的方式发送 Discover 请求报文，以发现网络中的 DHCP 服务器。所有收到 Discover 报文的 DHCP 服务器都会发送应答报文，DHCP 客户端据此可以知道网络中存在的 DHCP 服务器的位置
Offer(0x02)	DHCP 服务器收到 Discover 报文后，就会在所配置的地址池中查找一个合适的 IP 地址，加上相应的租约期限和其他配置信息(如网关、DNS 服务器等)，构造一个 Offer 报文，发送给 DHCP 客户端，告知用户本服务器可以为其提供 IP 地址，但这个报文只是告诉 DHCP 客户端可以提供 IP 地址，最终还需要客户端通过 ARP 来检测该 IP 地址是否重复

续表

报文种类	说　明
Request(0x03)	DHCP客户端可能会收到很多Offer请求报文，所以必须在这些应答中选择一个。通常选择第1个Offer应答报文的服务器作为自己的目标服务器，并向该服务器发送一个广播的Request请求报文，通告选择的服务器，希望获得所分配的IP地址。另外，DHCP客户端在成功获取IP地址后，在地址使用租期达到50%时，会向DHCP服务器发送单播Request请求报文请求续延租约，如果没有收到ACK报文，在租期达到87.5%时，则会再次发送广播的Request请求报文以请求续延租约
ACK(0x05)	DHCP服务器收到Request请求报文后，根据Request报文中携带的用户MAC地址来查找有没有相应的租约记录，如果有，则发送ACK应答报文，通知用户可以使用分配的IP地址
NAK(0x06)	如果DHCP服务器收到Request请求报文后，没有发现有相应的租约记录或者由于某些原因无法正常分配IP地址，则会向DHCP客户端发送NAK应答报文，通知用户无法分配合适的IP地址
Release(0x07)	当DHCP客户端不再需要使用分配的IP地址时(一般出现在客户端关机、下线等状况)就会主动向DHCP服务器发送Release请求报文，告知服务器用户不再需要分配IP地址，请求DHCP服务器释放对应的IP地址
Decline(0x04)	DHCP客户端收到DHCP服务器ACK应答报文后，通过地址冲突检测发现服务器分配的地址冲突或者由于其他原因导致不能使用，则会向DHCP服务器发送Decline请求报文，通知服务器所分配的IP地址不可用，以期获得新的IP地址
Inform(0x08)	DHCP客户端如果需要从DHCP服务器端获取更为详细的配置信息，则应向DHCP服务器发送Inform请求报文；DHCP服务器在收到该报文后，根据租约查找到相应的配置信息后，向DHCP客户端发送ACK应答报文。目前基本上不用了

DHCP报文格式如图3-10所示。

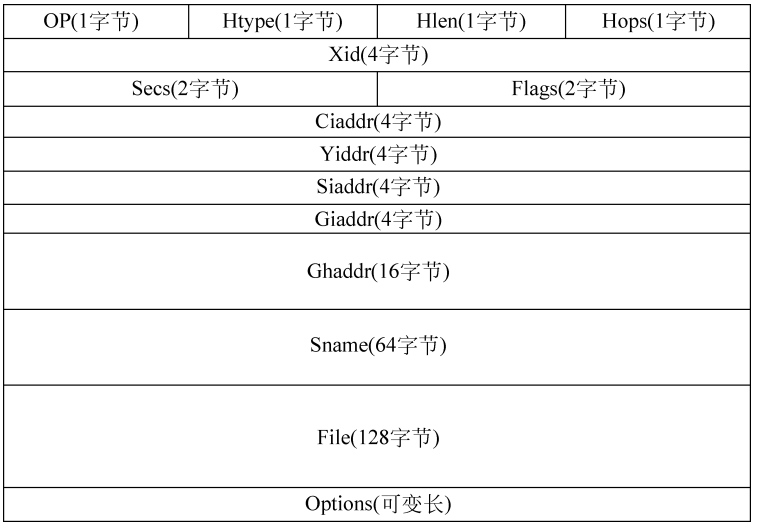

图3-10　DHCP报文格式

OP：报文的操作类型。分为请求报文和响应报文。1 表示请求报文，2 表示应答报文，即客户端传送给服务器的封包，设为 1，反之为 2。

请求报文：DHCP Discover、DHCP Request、DHCP Release、DHCP Inform 和 DHCP Decline。

应答报文：DHCP Offer、DHCP ACK 和 DHCP NAK。

Htype：DHCP 客户端的 MAC 地址类型。MAC 地址类型其实是指明网络类型。当 Htype 值为 1 时表示最常见的以太网 MAC 地址类型。

Hlen：DHCP 客户端的 MAC 地址长度。以太网 MAC 地址的长度为 6 字节，即以太网时 Hlen 的值为 6。

Hops：DHCP 报文经过的 DHCP 中继的数目，默认值为 0。DHCP 请求报文每经过一个 DHCP 中继，该字段就会增加 1。当没有经过 DHCP 中继时值为 0（若数据包需经过 router 传送，则每站加 1；若在同一网内，则为 0）。

Xid：客户端通过 DHCP Discover 报文发起一次 IP 地址请求时选择的随机数，相当于请求标识。用来标识一次 IP 地址请求过程。在一次请求中所有报文的 Xid 都是一样的。

Secs：DHCP 客户端从获取 IP 地址或者续约过程开始到现在所消耗的时间，以秒为单位。在没有获得 IP 地址前该字段始终为 0（DHCP 客户端开始 DHCP 请求后所经过的时间。目前尚未使用，固定值为 0）。

Flags：标志位，只使用第 0 比特位，是广播应答标识位，用来标识 DHCP 服务器应答报文是采用单播还是广播发送，0 表示采用单播方式发送，1 表示采用广播方式发送。其余位尚未使用，即从 0～15bits，最左 1bit 为 1 时表示服务器将以广播方式将封包传送给客户端。

需要注意的是，在客户端正式分配了 IP 地址之前的第 1 次 IP 地址请求过程中，所有 DHCP 报文都是以广播方式发送的，包括客户端发送的 DHCP Discover 和 DHCP Request 报文，以及 DHCP 服务器发送的 DHCP Offer、DHCP ACK 和 DHCP NAK 报文。当然，如果是由 DHCP 中继器转的报文，则都是以单播方式发送的。另外，IP 地址续约、IP 地址释放的相关报文都是采用单播方式进行发送的。

Ciaddr：DHCP 客户端的 IP 地址。仅在 DHCP 服务器发送的 ACK 报文中显示，因为在得到 DHCP 服务器确认前，DHCP 客户端还没有分配到 IP 地址。在其他报文中均显示，只有客户端是 Bound、Renew、Rebinding 状态，并且能响应 ARP 请求时，才能被填充。

Yiaddr：DHCP 服务器分配给客户端的 IP 地址。仅在 DHCP 服务器发送的 Offer 和 ACK 报文中显示，其他报文中显示为 0。

Siaddr：下一个为 DHCP 客户端分配 IP 地址等信息的 DHCP 服务器的 IP 地址。仅在 DHCP Offer、DHCP ACK 报文中显示，在其他报文中显示为 0（用于 Bootstrap 过程中的 IP 地址）。

一般来讲是服务器的 IP 地址，但是注意，根据 OpenWrt 源码给出的注释，当报文的源地址、Siaddr、Options-> server_id 字段不一致（有经过跨子网转发）时，通常认为 options-> server_id 字段为真正的服务器 IP，Siaddr 有可能是多次路由跳转中的某个路由的 IP。

Giaddr：DHCP客户端发出请求报文后经过的第1个DHCP中继的IP地址。如果没有经过DHCP中继，则显示为0(转发代理(网关)IP地址)。

Chaddr：DHCP客户端的MAC地址。在每个报文中都会显示对应DHCP客户端的MAC地址。

Sname：为DHCP客户端分配IP地址的DHCP服务器名称(DNS域名格式)。在Offer和ACK报文中显示发送报文的DHCP服务器名称，其他报文显示为0。

File：DHCP服务器为DHCP客户端指定的启动配置文件名称及路径信息。仅在DHCP Offer报文中显示，其他报文中显示为空。

Options：可选项字段，长度可变，格式为“代码＋长度＋数据”，如表3-27所示。

表3-27　DHCP Options可变长度及说明

代　码	长　度	说　明
1	4字节	子网掩码
3	长度可变，必须是4字节的倍数	默认网关(可以是一个路由器IP地址列表)
6	长度可变，必须是4字节的倍数	DNS服务器(可以是一个DNS服务器IP地址列表)
15	长度可变	域名称(主DNS服务器名称)
42	长度可变，必须是4字节的倍数	NTP服务器(可以是一个NTP服务器IP地址列表)
44	长度可变，必须是4字节的倍数	WINS服务器(可以是一个WINS服务器IP地址列表)
51	4字节	有效租约期(以秒为单位)
53	1字节	报文类型(1 ～ 8)分别表示：Discover、Offer、Request、Decline、ACK、NAK、Release、Inform
58	4字节	续约时间
60	长度可变	Authentication for DHCP Message，用来完成基于标准DHCP，以在客户端输入用户名和密码的方式进行地址鉴权，主要用在按用户认证收费场合，与之对应的是PPPOE认证计费
255	0	标记Options结束

3.7.3　DHCP工作流程

(1) IP地址分配方式，DHCP服务器负责接收客户端的DHCP请求，集中管理所有客户机的IP地址设定资料，并负责处理客户端的DHCP请求，相比于BOOTP，DHCP通过“租约”实现动态分配IP的功能，实现IP的时分复用，从而解决IP资源短缺的问题。

其地址分配方式有以下3种。

人工配置：由管理员对每台具体的计算机指定一个地址。

自动配置：服务器为第1次连接网络的计算机分配一个永久地址，DHCP客户端第1次成功地从DHCP服务器端分配到一个IP地址之后，就永远使用这个地址。

动态配置：在一定的期限内将地址租给计算机，客户端第1次从DHCP服务器分配到IP地址后，并非永久地使用该地址，每次使用后，DHCP客户端就得释放这个IP地址，并且

租期结束后客户必须续租或者停用该地址，而对于路由器，经常使用的地址分配方式是动态配置。

(2) 租约表，包含静态租约表和动态租约表。

静态租约表：对应一个静态租约存储文件，服务器运行时从文件中读取静态租约表。

动态租约表：对应一个周期存储文件，服务器周期性将租约表存进该文件，在程序开始时将会读取上次存放的租约表(租约表记录了当前所有分配的租约，包括静态链接的)。

DHCP 服务器一直处在被动接受请求的状态，当有客户端请求时，服务器会读取客户端当前所在的状态及客户端的信息，并在静态租约表和动态租约表中进行检索，以便找到相应的表项，再根据客户端的状态执行不同的回复。

当收到客户端的首次请求时，DHCP 服务器先查找静态租约表；若存在请求的表项，则返回这个客户的静态 IP 地址，否则从 IP 地址池中选择可用的 IP 分配给客户，并将信息添加到动态数据库中。此外，服务器将会周期性地刷新租约表并写入文件存档，在这个过程中会顺便对动态租约表进行租期检查。

(3) 工作流程，可分为客户端流程和服务器端流程。

客户端登录网络：

客户机初始化 & 寻找 DHCP 服务器(DHCP Discover)：DHCP 客户端启动时，计算机发现本机上没有任何 IP 地址设定，将以广播方式通过 UDP 67 端口发送 DHCP Discover 发现信息来寻找 DHCP 服务器，因为客户机还不知道自己属于哪一个网络，所以封包的源地址为 0.0.0.0，目的地址为 255.255.255.255，向网络发送特定的广播信息。网络上每台安装了 TCP/IP 的主机都会接收这个广播信息，但只有 DHCP 服务器才会作出响应。

DHCP Discover 的等待时间预设为 1s，也就是当客户机将第 1 个 DHCP Discover 封包送出去之后，在 1s 之内没有得到回应，就会进行第 2 次 DHCP Discover 广播。若一直没有得到回应，客户机就会将这一广播包重新发送 4 次(以 2、4、8、16s 为间隔，加上 1～1000ms 随机长度的时间)。如果都没有得到 DHCP 服务器的回应，客户机则会从 169.254.0.0/16 这个自动保留的私有 IP 地址中选用一个 IP 地址，并且每隔 5min 重新广播一次，如果收到某个服务器的响应，则继续 IP 租用过程。

分配 IP 地址 & 提供 IP 地址租用(DHCP Offer)：DHCP 服务器收到客户端发出的 DHCP Discover 广播后，通过解析报文，查询 dhcpd.conf 配置文件。它会从那些还没有租出去的地址中选择最前面的空置 IP，连同其他 TCP/IP 设定，通过 UDP 68 端口响应给客户端一个 DHCP Offer 数据包(包中包含 IP 地址、子网掩码、地址租期等信息)。告诉 DHCP 客户端，该 DHCP 服务器拥有资源，可以提供 DHCP 服务。

此时还是使用广播进行通信，源 IP 地址为 DHCP 服务器的 IP 地址，目标地址为 255.255.255.255。同时，DHCP 服务器为此客户端保留它提供的 IP 地址，从而不会为其他 DHCP 客户分配此 IP 地址。

由于客户端在开始时还没有 IP 地址，所以在其 DHCP Discover 封包内会带有其 MAC 地址信息，并且有一个 XID 编号来辨别该封包，DHCP 服务器响应的 DHCP Offer 封包则

会根据这些资料传递给要求租约的客户。

接受 IP 地址 & 接受 IP 租约(DHCP Request)：DHCP 客户端接收到 DHCP Offer 提供的信息后，如果客户机收到网络上多台 DHCP 服务器的响应，则一般接受最先到达的那个，然后以广播的方式回答一个 DHCP Request 数据包(包中包含客户端的 MAC 地址、接受的租约中的 IP 地址、提供此租约的 DHCP 服务器地址等)。告诉所有 DHCP 服务器它将接受哪一台服务器提供的 IP 地址，所有其他的 DHCP 服务器撤销它们提供的信息，以便将 IP 地址提供给下一次 IP 租用请求。

此时，由于还没有得到 DHCP 服务器的最后确认，所以客户端仍然使用 0.0.0.0 作为源 IP 地址，使用 255.255.255.255 作为目标地址进行广播。

事实上，并不是所有的 DHCP 客户端都会无条件地接受 DHCP 服务器的 Offer，特别是如果这些主机上安装有其他 TCP/IP 相关的客户机软件。客户端也可以用 DHCP Request 向服务器提出 DHCP 选择，这些选择会以不同的号码填写在 DHCP Option Field 里面。客户机可以保留自己的一些 TCP/IP 设定。

IP 地址分配确认 & 租约确认(DHCP ACK)：当 DHCP 服务器接收到客户机的 DHCP Request 之后，会广播返给客户机一个 DHCP ACK 消息包，表明已经接受客户机的选择，告诉 DHCP 客户端可以使用它提供的 IP 地址，并将这一 IP 地址的合法租用及其他配置信息都放入该广播包并发给客户机。

客户端在接收到 DHCP ACK 广播后，会向网络发送 3 个针对此 IP 地址的 ARP 解析请求以执行冲突检测，查询网络上有没有其他机器使用该 IP 地址；如果发现该 IP 地址已经被使用，客户机则会发出一个 DHCP Decline 数据包给 DHCP 服务器，拒绝此 IP 地址租约，并重新发送 DHCP Discover 信息。此时，在 DHCP 服务器管理控制台中，会将此 IP 地址显示为 BAD_ADDRESS。

如果网络上没有其他主机使用此 IP 地址，则客户机的 TCP/IP 会使用租约中提供的 IP 地址完成初始化，从而可以和其他网络中的主机进行通信。

客户端重新登录：以后 DHCP 客户端每次重新登录网络时，就不需要再发送 DHCP Discover 发现信息了，而是直接发送包含前一次所分配的 IP 地址的 DHCP Request 请求信息。当 DHCP 服务器收到这一信息后，它会尝试让 DHCP 客户机继续使用原来的 IP 地址，并回答一个 DHCP ACK 确认信息。如果此 IP 地址已无法再分配给原来的 DHCP 客户端使用，则 DHCP 服务器会给 DHCP 客户端回答一个 DHCP NACK 否认信息。当原来的 DHCP 客户机收到此 DHCP NACK 否认信息后，它就必须重新发送 DHCP Discover 发现信息来请求新的 IP 地址。

客户端更新租约：DHCP 服务器向 DHCP 客户机出租的 IP 地址一般有一个租借期限，期满后 DHCP 服务器便会收回出租的 IP 地址。如果 DHCP 客户机要延长其 IP 租约，则必须更新其 IP 租约。

客户端会在租期过去 50%时，直接向为其提供 IP 地址的 DHCP 服务器发送 DHCP Request 消息包。如果客户端接收到该服务器回应的 DHCP ACK 消息包，客户端就会根据

包中所提供的新的租期及其他已经更新的 TCP/IP 参数，更新自己的配置，IP 租用更新完成。如果没有收到该服务器的回复，则客户端继续使用现有的 IP 地址，因为当前租期还有 50%。

如果在租期过去 50%时没有更新，则客户端将在租期过去 87.5%时再次与为其提供 IP 地址的 DHCP 联系。如果还不成功，到租约的 100%时，客户端则必须放弃这个 IP 地址，重新申请。如果此时无 DHCP 可用，客户端则会使用 169.254.0.0/16 中随机的一个地址，并且每隔 5min 再进行尝试。

服务器处理流程：

1. DHCP Offer

静态租用：首先匹配 MAC 地址，看是否能在静态租约表中找到对应的项，若能找到就把 IP 分配给它。静态表中的 IP 不能被其他客户使用。

动态租用：服务器试图分配给客户端上次分配过的 IP，在这之前检查这个 IP 是否正在使用。

当 DHCP Discover 中含有 Request IP 时，检查该 IP 是否在地址池范围，是否正在使用，是否到期，是否是静态 IP，以及网络上是否已经存在。

DHCP Discover 不含 Request IP，从地址池上寻找一个最小的可用 IP 分配。

2. DHCP ACK

根据是否含有 Request IP 和 Server IP 识别客户端现在处于 init_reboot、selecting、renewing、rebinding 中的哪种状态，并根据以下规则执行 DHCP ACK 回复：

若客户端处于 selecting 状态，则验证 Request IP 和 Server IP 是否同服务器中的匹配。

若客户端处于 init_reboot 状态，则验证 Request IP 是否符合租约记录。

若客户端处于 renewing/rebinding 状态，则验证 Client IP 是否符合租约记录。

3. DHCP NAK

请求的 IP 是静态 IP，但是 MAC 地址无法与其对应。上面 DHCP ACK 中验证失败，服务器还可能会收到其他包。

4. DHCP Decline

服务器会把租约表中相关客户端硬件地址置空，并保存这个地址一段时间。

5. DHCP Release

清空租期回收 IP。

6. DHCP Inform

回复 DHCP ACK，数据包含关于服务器的信息。

3.7.4 DHCP 配置文件

在主配置文件 dhcpd.conf 中，可以使用声明、参数、选项这 3 种类型进行配置，各自的作用和表现形式如下。

声明：用来描述 dhcpd 服务器中对网络布局的划分，是网络设置的逻辑范围。常见的

声明是 subnet、host，其中 subnet 声明用来约束一个网段。host 声明用来约束一台特定主机。

参数：由配置关键字和对应的值组成，总是以“；”(分号)结束，一般位于指定的声明范围内，用来设置所在范围的运行特性(如默认租约时间、最大租约时间等)。

选项：由 option 引导，后面跟具体的配置关键字和对应的值，也是以“；”结束，用于指定分配给客户机的各种地址参数(如默认网关地址、子网掩码、DNS 服务器地址等)。

为了使配置文件的结构更加清晰、全局配置通常会放在配置文件 dhcpd.conf 的开头部分，可以是配置参数，也可以是配置选项。常用的全局配置参数和选项如下。

ddns-update-style：动态 DNS 更新模式。用来设置与 DHCP 服务相关联的 DNS 数据动态更新模式。在实际的 DHCP 应用中很少用到该参数。将值设为 none 即可。

default-lease-time：默认租约时间。单位为秒，表示客户端可以从 DHCP 服务器租用某个 IP 地址的默认时间。

max-lease-time：最大租约时间。单位为秒，表示允许 DHCP 客户端请求的最大租约时间，当客户端未请求明确的租约时间时，服务器将采用默认租约时间。

option domain-name：默认搜索区域。为客户机指定解析主机名时的默认搜索域，该配置选项将体现在客户机的/etc/resolv.conf 配置文件中，如 search benet.com。

option domain-name-servers：DNS 服务器地址。为客户端指定解析域名时使用的 DNS 服务器地址，该配置选项同样将体现在客户机的/etc/resolv.conf 配置文件中，如 nameserver 202.106.0.20。当需要设置多个 DNS 服务器地址时，以逗号进行分隔。

一台 DHCP 服务器可以为多个网段提供服务，因此 subnet 网段声明必须有而且可以有多个。例如，若要 DHCP 服务器为 192.168.100.0/24 网段提供服务，用于自动分配的 IP 地址范围为 192.168.100.100～192.168.100.200，如果为客户机指定默认网关地址为 192.168.100.254，则可以修改 dhcpd.conf 配置文件，参考以下内容调整 subnet 网段声明：

```
vim /etc/dhcp/dhcpd.conf <! -- 编辑主配置文件 -->

subnet 192.168.100.0 netmask 255.255.255.0 {    <! -- 声明网段地址 -->
    range 192.168.100.100 192.168.100.200;      <! -- 设置地址池，可以有多个 -->
    option routers 192.168.100.254;             <! -- 指定默认网关地址 -->
}
```

host 声明用于设置单个主机的网络属性，通常用于为网络打印机或个别服务器分配固定的 IP 地址(保留地址)，这些主机的共同特点是要求每次获取的 IP 地址相同，以确保服务的稳定性。

host 声明通过 host 关键字指定需要使用保留地址的客户机名称，并使用 hardware ethernet 参数指定该主机的 MAC 地址，使用 fixed-address 参数指定保留给该主机的 IP 地址。例如，若要为打印机 prtsvr(MAC 地址为 00：0C：29：0D：BA：6B)分配固定的 IP 地址 192.168.100.101，则可以修改 dhcpd.conf 配置文件，参考以下内容在网段声明内添加

host 主机声明。

在 Windows 系统中查看 MAC 地址，命令如下：

```
getmac

物理地址                    传输名称
============================================================
00-0C-29-0D-BA-6B      \Device\Tcpip_{92E3F48B-40F0-4A0D-9604-6386AAAE3233}
```

在 Linux 系统中查看 MAC 地址，命令如下：

```
ip a

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 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
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default
qlen 1000
    link/ether 00:0c:29:ea:c5:08 brd ff:ff:ff:ff:ff:ff
    inet 172.16.23.182/24 brd 172.16.23.255 scope global noprefixroute dynamic ens33
       valid_lft 1758sec preferred_lft 1758sec
    inet6 fe80::237d:375e:cce2:d817/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: ens37: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default
qlen 1000
    link/ether 00:0c:29:ea:c5:12 brd ff:ff:ff:ff:ff:ff
4: ens38: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default
qlen 1000
    link/ether 00:0c:29:ea:c5:1c brd ff:ff:ff:ff:ff:ff
```

DHCP 配置固定 IP，命令如下：

```
vim /etc/dhcp/dhcpd.conf

host win7 {
  hardware ethernet 00:0C:29:0D:BA:6B;     <! -- 客户机的 MAC 地址 -- >
  fixed-address 192.168.100.101;           <! -- 分配给客户机的 IP 地址 -- >
}
```

3.7.5 启动 dhcpd 服务

在启动 dhcpd 服务之前，应确认提供 DHCP 服务器的网络接口具有静态指定的固定 IP 地址，并且至少有一个网络接口的 IP 地址与 DHCP 服务器中的一个 subnet 网段相对应，否则将无法正常启动 dhcpd 服务。例如，DHCP 服务器的 IP 地址为 192.168.100.10，用于为网段 192.168.100.0/24 内的其他客户机提供自动分配地址服务。

安装 DHCP 软件包以后，对应的系统服务脚本位于/usr/lib/systemd/system/dhcpd.service，可以使用 systemd 服务进行控制。例如，执行以下操作可以启动 dhcpd 服务，并检查 UDP 的 67 端口是否在监听，以确认 DHCP 服务器是否正常。

```
systemctl start dhcpd        <! -- 启动 dhcp 服务 -- >

systemctl enable dhcpd       <! -- 将服务设置为开机自动启动 -- >

netstat -anptu | grep 67     <! -- 监听 DHCP 服务器端口号 -- >
udp          0          0 0.0.0.0:67          0.0.0.0:*           2102/dhcpd
udp          0          0 0.0.0.0:67          0.0.0.0:*           1064/dnsmasq
```

查看 dhcpd 服务，命令如下：

```
systemctl status dhcpd
```

停止 dhcpd 服务，命令如下：

```
systemctl stop dhcpd
```

重启 dhcpd 服务，命令如下：

```
systemctl restart dhcpd
```

查看 dhcpd 日志，命令如下：

```
tail -300f /var/log/messages
```

3.7.6　虚拟机测试 DHCP Server

Windows 系统中 VMware 有一个虚拟网络编辑器，可以直接关闭 DHCP。

Windows 获取 IP 只要到网络配置界面选择“自动获取 IP”即可，也可以使用命令进行操作，示例如下：

```
ipconfig /renew        <! -- 可以为主机重新获取新的 IP 地址 -- >
ipconfig /release      <! -- 释放 IP 地址 -- >
tracert IP 地址        <! -- 可以测试从当前主机到目的主机经过的网络节点 -- >
route print            <! -- 查看路由表 -- >
```

使用 macOS VMware Fusion 专用网络关闭 DHCP，命令如下：

```
cd /Library/Preferences/VMware\ Fusion
sudo vim networking
```

按 I 键进入编辑，修改完成后，按 Esc 键，输入“：wq”退出编辑。查看内容如下：

```
VERSION = 1,0
answer VNET_1_DHCP no
answer VNET_1_DHCP_CFG_HASH D86B852E44E830A664C32C059E594C4E62E7177B
answer VNET_1_HOSTONLY_NETMASK 255.255.255.0
answer VNET_1_HOSTONLY_SUBNET 172.16.211.0
answer VNET_1_VIRTUAL_ADAPTER yes
answer VNET_8_DHCP yes
answer VNET_8_DHCP_CFG_HASH FDD8224F31C72F88211DCE6F2199B4FFE3961CF8
answer VNET_8_HOSTONLY_NETMASK 255.255.255.0
answer VNET_8_HOSTONLY_SUBNET 172.16.23.0
answer VNET_8_NAT yes
answer VNET_8_VIRTUAL_ADAPTER yes
add_bridge_mapping en0 2
add_bridge_mapping en7 3
```

把上面的 DHCP 修改为 no 就可以了。

如果不知道修改哪一个，则可直接看 VNET 后面有没有带 NAT 的那一组，没有带 NAT 的那组使用的就是专用网络，然后修改并保存后，重启 VMware 和虚拟机才能生效。

在 Linux 客户机中可以设置使用 DHCP 的方式获取地址。只需编辑对应网卡的配置文件，修改或添加 BOOTPROTO = dhcp 配置行，并重新加载配置文件或者重新启动 Network 服务。例如，执行以下操作可修改网卡配置文件，并重新加载配置以通过 DHCP 方式自动获取地址：

```
vim /etc/sysconfig/network - scripts/ifcfg - ens32

TYPE = Ethernet
PROXY_METHOD = none
BROWSER_ONLY = no
BOOTPROTO = dhcp
DEFROUTE = yes
NAME = ens32
DEVICE = ens32
ONBOOT = yes
```

重启网卡及网络，命令如下：

```
ifdown ens32 ; ifup ens32
systemctl restart network
```

在 Linux 客户机中，还可以使用 dhclient 工具来测试 DHCP 服务器。若直接执行 dhclient 命令，则 dhclient 将尝试为除回环接口 lo 以外的所有网络接口通过 DHCP 方式申请新的地址，然后自动转入后台继续运行。当然，测试时可以指定一个具体的网络接口，并结合-d 选项使其在前台运行，测试完毕后按快捷键 Ctrl＋C 终止。例如，执行 dhclient -d ens32 命令后，可以为网卡 ens32 自动获取新的 IP 地址，并显示获取过程，命令如下：

```
dhclient  - d ens32
```

所有网卡获取 IP，命令如下：

```
dhclient
```

指定网卡获取 IP，命令如下：

```
dhclient ens32
```

所有网卡释放 IP，命令如下：

```
dhclient -r
```

指定网卡释放 IP，命令如下：

```
dhclient -r ens32
```

ifconfig eth0 up/down 与 ifup/ifdown eth0 的区别如下。

（1）相同点：ifconfig 网络接口名 up 命令用于启动网络接口，等同于 ifup；ifconfig 网络接口名 down 命令用于停用网络接口，等同于 ifdown。

（2）不同点：ifconfig 在配置文件/etc/sysconfig/network-scripts/ifcfg-ethx 中 DEVICE＝eth0 时，使用 ifconfig eth0 up/down 才会有效，如果 DEVICE＝eth1，则再使用 ifconfig eth0 up/down 便会出现“eth0：unknown interface：没有那个设备”错误。

ifup 与 ifdown 程序其实是脚本而已，它们会直接到 /etc/ sysconfig/network-scripts 目录下搜索对应的配置文件，例如 ifcfg-eth0，它会找出 ifcfg-eth0 文件的内容，然后加以设置。在 ifcfg-eth0 文件中，如果 DEVICE＝eth1，则使用 ifup/ifdown eth0 或 ifup/ifdown eth1 都能启动或关闭 eth1 网络设备。

3.8　多种开发语言的组合开发介绍

目前主流的几种开发语言能够完成开发任务，并且具有成熟的框架和生态圈，这里通过比较语言简洁性、运行平台的依赖性、框架和生态的成熟性、业务的场景，选择 Go 作为后端主要开发语言，其中 Shell Script 作为系统任务和外部服务调用的开发语言；前端采用 Vue.js 作为开发语言，其主要版本信息如表 3-28 所示。

表 3-28　开发语言选型及说明

语　言	环　境	版　本	部　署
Go	CentOS	CentOS 7.x Go 1.19	编译部署
Shell Script	CentOS	CentOS 7.x 内核 3.10＋	加密部署

续表

语　言	环　境	版　本	部　署
Vue.js	Nginx	Nginx 1.12+ Vue 4.5+	编译部署

作为后端的主要开发语言 Go，也是一门比较新的开发语言，它拥有众多特性和优势，并且具有完善的生态。

(1) Go 语言官方介绍：Go 语言是由谷歌推出的一门编程语言。Go 是一个开源的编程语言，它能够让开发构造简单、可靠且高效的软件变得容易。

(2) Go 语言主要开发者：Go 语言始于 2007 年，当时只是谷歌内部的一个项目，其最初设计者是 Rebert Griesemer、UNIX 泰斗 Rob Pike 和 Ken Thompsopn。2009 年 11 月 10 日，Go 语言以一个自由的开源许可方式公开亮相。

Go 语言由其原始设计者加上 Russ Cox、Andrew Gerrand、Ian Lance Taylor 及其他许多人在内的一个谷歌团队开发。Go 语言采取一种开放的开发模式，吸引了许多来自世界各地的开发者为这门语言的发展贡献力量，其中有些开发者获得了非常好的声望，因此他们也获得了与谷歌员工一样的代码提交权限。

(3) Go 语言在最近两年比较显而易见的变化主要包含以下方面。①本身的自举：也就是说，Go 语言几乎用 Go 语言程序重写了自己，仅留有一些汇编程序。Go 语言的自举非常彻底，包括了最核心的编译器、链接器、运行时系统等。现在任何学习 Go 语言的人都可以直接读它的源代码了。此变化也使用 Go 程序的跨平台编译变得轻而易举；②运行时系统的改进：这主要体现在更高效的调度器、内存管理及垃圾回收方面。调度器已能让 goroutine 更及时地获得运行机制。运行时系统对内存的利用和控制也更加精细了。因垃圾回收而产生的调度停顿时间已经小于原来的 1%。另外，最大 P 数量的默认值由原先的 1 变为与当前计算机的 CPU 核心数相同；③标准工具的增强：在 Go 1.4 加入 go generate 之后，一个惊艳的程序调试工具 go tool trace 也被添加进来了。另外，go tool compile、go tool asm 和 go tool link 等工具也已到位；一旦安装好 Go，就可以直接使用它们。同时，绝大多数的标准库工具和命令得到了不同程度的改进；④访问控制的细化：这种细化始于 Go 1.4，正式支持始于 Go 1.5，至今已被广泛应用。经过细化，对于 Go 程序中的程序实体，除了原先的两种访问控制级别(公开和包级私有)之外，又多了一种模块级私有。这是通过把名称首字母大写的程序实体放入 internal 代码包实现的；⑤vendor 机制的支持：自 Go 1.5 之后，一个特殊的目录 vendor 被逐渐启用。它用于存放其父目录中的代码包所依赖的那些代码包。在程序被编译时，编译器会优先引用存于其中的代码包，这为固化程序的依赖代码迈出了很重要的一步。在 Go 1.7 中，vendor 目录及其背后的机制被正式支持。

当然，上述变化并不是全部。它的标准库也经历了超多的功能和性能改进。

(4) Go 语言的发展方向主要为网络编程语言、区块链开发领域、高性能分布式系统领域。

(5) Go 语言特性包含以下方面。①开放源代码：这显示了 Go 作者开放的态度及营造

语言生态的决心。顺便说一句，Go 本身就是主要用 Go 语言编写的；②静态类型和编译型：在 Go 中，每个变量或常量都必须在声明时指定类型，并且不可改变。另外，程序必须通过编译生成归档文件或可执行文件，而后才能被使用或执行。不过，其语法非常简单，就像一些解释性脚本语言那样，易学易用；③跨平台：这主要是指跨计算架构和操作系统。目前，它支持绝大部分主流的计算架构和操作系统，并且这个范围还在不断扩大。只要下载与之对应的 Go 语言安装包，并且经过简单的安装和设置，就可以使用 Go 了。除此之外，在编写 Go 语言程序的过程中，几乎感觉不到不同平台的差异；④自动垃圾回收：程序在运行过程中的垃圾回收工作一般由 Go 运行时系统全权负责。不过，Go 也允许对此项工作进行干预；⑤原生的并发编程：拥有自己的并发编程模型，其主要组成部分有 goroutine(也可称为 Go 例程)和 channel(也可称为通道)。另外，还拥有一个特殊的关键字 go；⑥完善的构建工具：它自带了很多强大的命令和工具，通过它们，可以很轻松地完成 Go 程序的获取、编译、测试、安装、运行、分析等一系列工作；⑦多编程范式：Go 支持函数式编程。函数类型为第一等类型，可以方便地传递和赋值。此外，它还支持面向对象编程，有接口类型与实现类型的概念，但使用嵌入替代了继承；⑧代码风格强制统一：Go 安装包中有一个自己的代码格式化工具，可以用来统一程序的编码风格；⑨高效的编程和运行：Go 简单，直接的语法可以快速编写程序。加之它拥有更强大的运行时系统，程序可以充分利用计算环境飞快运行；⑩丰富的标准库：Go 是通用的编程语言，其标准库中有很多开箱即用的 API。尤其是在编写系统级程序、Web 程序和分布式程序时，几乎无须依赖第三方库。

3.9　匿名链路的测试节点信息配置

链路节点的核心作用是承载真实用户的流量，并充当真实用户访问服务请求，可以避免真实用户直接暴露在外面，同时有些服务在用户所在地区无法直接访问，亦可通过节点流量转发进行实现。节点服务器本身不需要太高的配置，也不参与 IO 计算，主要作用是充当路由器，本书测试所采用的测试节点信息如表 3-29 所示，为了避免操作系统本身的漏洞，节点系统需支持多种，本书支持了 3 种主流的操作系统，分别是 CentOS、Ubuntu、Debian。

表 3-29　测试节点信息

云服务商	国家/地区	数　量	配　置	时　间	价　格	系　统
阿里云	华北	2	1H1G20G	6 个月	53 元/月	CentOS 7.9
阿里云	香港	2	2H1G20G	6 个月	71 元/月	CentOS 7.9
阿里云	硅谷	2	2H1G20G	6 个月	59 元/月	CentOS 7.9
腾讯云	上海	1	2H2G50G	6 个月	77 元/月	CentOS 7.9
华为云	新加坡	1	1H1G40G	6 个月	98 元/月	CentOS 7.9
亚马逊云	北京	1	1H1G20G	6 个月	37 元/月	CentOS 7.9
合计		9		6 个月	3468 元	